INFORMATION
SYSTEMS IN
MANAGEMENT

Information Systems in Management (2012) Vol. 1 (1) 25−37

BUSINESS INTELLIGENCE AND NOSQL DATABASES

JERZY DUDA

Department of Applied Computer Science, Faculty of Management, AGH University of Science and Technology (AGH)

NoSQL databases become more and more popular, not only in typical Internet applications. They allow to store large volumes of data (so called big data), while ensuring fast retrieving and fast appending. The main disadvantage of NoSQL databases is that they do not use relational model of data and usually do not offer any declarative query language similar to SQL. This raises the question how NoSQL databases can be used for OLAP processing and other Business Intelligence tasks. In the paper the author presents the most common types of NoSQL databases, describes MapReduce paradigm and discusses models of OLAP processing for such databases. Finally some preliminary results of aggregation performance in non-relational environment are presented.

Keywords: Business Intelligence, Databases, NoSQL, Big data, OLAP

1. Introduction

Since E. Cobb in 1970 proposed relational model of databases it has become a dominant standard for data storage in information systems up to present. In this period some other models have been proposed, including object oriented and XML-based ones. Despite these new models had some advantages over the traditional relational model they were implemented rather as experimental projects and nowadays they are developed mainly by the open source community. Large relational database management systems (RDBMS) vendors like Oracle and Microsoft have incorporated those new ideas into their relational systems. Currently their

databases natively support XML data types, MS SQL Server allows for creation CLR types that can be used for defining objects, and Oracle allows for defining objects in its PL/SQL language (what makes Oracle RDBMS in fact object-relational database management system). There is no doubt that relational model owes its popularity also to the universal, simple, but very powerful SQL language. This language, originally developed in IBM, allows a wide range of people, including non-programmers like managers, analysts and other decision makers to easy interact with data collected in databases.

The originally intended purpose of relational databases and the SQL language was to store data from various transactions occurred in business like orders, bills, salaries and make queries to these data. Further development of management information systems together with the development of computer hardware itself, on the one side, and the development of strategic and operational management as a response to higher and higher competition on the global market, on another side, contributed to the extensive use of RDBMS also in analytical processing. Data warehouses which form the basis of online analytical processing (OLAP) can exclusively or partially operate on the same relational model (Relational OLAP - ROLAP) and offer dedicated tools to easily transform data from relational databases into desired OLAP structures during extract, transform and load (ETL) processes. The details of various OALP architectures will be discussed in chapter 3.

During the last decade an intensive development of Internet applications could be observed. Such platforms like large e-shops or social portals need to process often huge amount of data from millions of users in almost real time. This led to the development of completely new models of databases. Those models are collectively referred to as NoSQL (from "Not only SQL"). Their main goal is to store large quantities of data (so called big data) in a distributed, no-relational system to ensure fast access and append operations as well as a fault tolerance. Thus their main purpose is similar to the primary purpose of the relational databases – storing transaction data. As currently there is a strong demand from business side for OLAP and other Business Intelligence solutions, the question arises how NoSQL databases can be used in this kind of applications?

The main goal of the paper is to show whether, and if yes, how various types NoSQL databases can be applied for the analytical processing known from contemporary Business Intelligence suites. The paper is organized as follows. In the second chapter the idea of NoSQL databases is presented together with their taxonomy and MapReduce paradigm, as the most effective method of data processing for large databases. Next chapter is devoted to the idea of Business Intelligence, data warehouses and their architecture in typical environments that are based on traditional RDBMS. The author analyses how existing BI systems can deal with NoSQL databases, discusses pros and cons of the solutions, and presents some new

concepts in this field. Finally, the author presents some results of experiments with aggregation queries typical for OLAP and possible ways of their improvement.

2. NoSQL databases and MapReduce paradigm

Term "NoSQL" database was first used by C. Strozzi in 1998 for his database system that did not provide SQL interface. According to his definition "NoSQL is a fast, portable, relational database management system without arbitrary limits, (other than memory and processor speed) that runs under, and interacts with, the UNIX Operating System" [1]. Despite not utilizing SQL language this system was in fact a relational database system, but data were stored in ASCII files and could be manipulated by regular Unix tools instead of SQL. The "NoSQL" term has been rediscovered eleven years later by, among the others, E. Evans [2] who used this term in his blog while discussing open source databases of other types than relational. In this sense NoSQL databases can be seen as "non-relational, distributed, open-source and horizontally scalable" [3]. Thus original sense of "NoSQL" term meaning a relational database system without SQL language has change to a non-relational database system that is not like a typical SQL database, so the term should now be referred to as "Not only SQL".

There is no single model of NoSQL database. In fact, any database system that satisfies the conditions presented above (at least most of them including primarily to be non-relational) can be classified to NoSQL family. Nosql-database.org portal lists the following categories of NoSQL databases [3]:

- column store,
- document store,
- key-value store,
- graph databases,
- object oriented databases,
- XML databases,
- others like mutlimodel, multidimensional, and mutlivalue databases.

However, many sources regard only the first four as true NoSQL databases, as they are based on new concepts and are the most popular ones. In this paper the author will focus mainly on column store and document store models, as the databases utilizing these models are the most important from a business point of view and are the most widely used in various applications. Finally, the MapReduce programming model for effective processing of large databases will be discussed.

2.1. Column-oriented model

In the relational database management systems the data is stored typically in rows (tuples). In contrast, a column-oriented database engine (called CDBMS or

simply a column store) writes each table attribute in a separate column. For example a simple table of customer tuples presented in Fig. 1 could be written in the format presented in Fig. 2.

ID	Name	Address	City	Phone
1	A Glass Enterprise	212 S Tower	Centralia	1 360-736-1141
2	A-1 Mobile	389 Raubuck Rd	Winlock	1 360-262-0216
3	Abitz	W 181 Fredson Rd	Shelton	360-426-1928
4	Agnew Lumber Co.	PO Box 579	Centralia	360-736-8211

Figure 1. A simple table of customers in the relational model

This, of course, is a simplified model. In real implementations more complex structures are used, including mixed row-column ones (e.g. in C-Store database), data are usually compressed (as there is a lot of redundancy), and various indexes, caching and other techniques are applied to speed up queries.

ID	1,2,3,4
Name	A Glass Enterprise, A-1 Mobile, Abitz, Agnew Lumber Co.
Address	212 S Tower, 389 Raubuck Rd, W 181 Fredson Rd, PO Box 579
City	Centralia, Winlock, Shelton, Centralia
Phone	1 360-736-1141, 1 360-262-0216, 360-426-1928, 360-736-8211

Figure 2. A simple column store for customers' data

The idea of column-oriented data storage is not new and dates back to 1970s, when the research on transposed files and vertical partitioning was first published [4]. The first widely used system that utilized such a model was RAPID system built for Statistics Canada in 1976. A decade later the advantage of the decomposed storage model (DSM) over the row-wise storage (NSM) has been shown [5]. Nevertheless for many years the only one column-oriented database system commercially available was Sybase IQ. The situation has changed in a recent few years as many open source as well as commercial products have been released. The most popular open source projects that are able to store data in columns (many of them are also capable to store data in rows) are:

- Apache Cassandra – initially developed in Facebook,
- C-Store – first comprehensive design description of column store by researchers from few American universities, including MIT,
- HBase – running on the top of Hadoop; currently used for Facebook messaging system,
- Google Bigtable – data are indexed by triples <row, column, timestamp>, but tables are stored on the basis of column families.

The main advantages of column store systems over traditional RDBMS include first of all [6]:

- improved bandwith utilization, as access is necessary only for tables with those attributes that are required by a query,
- improved data compression, as attribute values from the same domain compress better than for mixed type values,
- improved code pipelining (instructions per cycle), as data can be accessed directly in an iterative way, not through tuple-based interface,
- improved cache storage, as caching only chosen attribute tables requires less space than caching whole tuples.

The main disadvantage of column stores is the cost of tuple reconstruction. The higher number of attributes is engaged in a query the slower is the answer. This, however, can be improved by using in-memory cache or by utilizing SSD drives in database servers. It is also worth to notice that OLAP processing and other Business Intelligence methods usually require a limited number of attributes in a time.

2.2. Document-oriented model

Databases storing user data in the form of documents are the flagship of the NoSQL family and are known simply as document stores. Although documents in such databases can be written in almost any format (e.g. Word or PDF), the highest number of possibilities is offered by semi-structured formats like XML, YAMSL, JSON (or its binary version BSON). Although databases using XML documents first appeared on the market (the are referred to as native XML databases), but nowadays the most popular document stores use JSON (JavaScript Object Notation) as a document format, as its structure is significantly less complex than XML and the documents occupy less storage space. The most popular JSON/BSON stores are: CouchDB, MongoDB and OrientDB. Fig. 3 presents a record from the table of customers that was considered earlier written in a simple JSON format.

```
{  "id": 1,
   "name": "A Glass Enterprise",
   "address": { "street": " 212 S Tower", "city": "Centralia"},
   "phone": " 1 360-736-1141" }
```

Figure 3. JSON record containing customer data. Author's own preparation

In this case the data is stored in tuples, like in the relational database, but in less rigid format. As values are always stored together with their key, the structure of rows can be changed in successive rows. A "schema-free" model allows to easily adjust a database to the changing information needs of business analytics.

The other advantages of document stores include:

- rich data structures – document in a store can represent objects or arrays, thus no ORM (object relational mapping) is required,
- better performance – data are usually stored in a single table or they are joined in memory, so traditional, time-consuming JOINS are unnecessary and usually not supported,
- horizontally scalable – document store works similar to distributed hash tables, and it can be scaled easily for a large number of nodes.

The main disadvantage of document stores is that they usually do not provide any declarative language for data manipulation (only OrientDB has SQL, but without support for JOIN clauses). Data processing requires to use some procedural language, so programming skills are needed in order to process data collected in such databases. However, this lets to write more complex queries that operate on a single row (like cursors in RDBMS). This ability can be applied to Business Intelligence tools for developing new kind of analysis or to improve existing ones.

2.3. MapReduce framework

Although MapReduce is not a database model, it plays a very important role in today's NoSQL databases. MapReduce has been developed at Google as a "programming model and an associated implementation for processing and generating large data sets" [7]. Although the idea is not entirely new, simple and easy to use MapReduce framework (sometimes called a paradigm) in recent years has become a new phenomenon in processing huge amount of data in a distributed environment. To use the framework two functions have to be defined: the map function, responsible for mapping key-values pairs coming from input data into some other key-values pairs:

$$\mathtt{Map(K,\ V)} \rightarrow \mathtt{list\ (K',\ V')},$$

and the reduce function, responsible for reducing values (first they are sorted and grouped on the basis of new keys), and producing the final output:

$$\mathtt{Reduce(K',\ list(V'))} \rightarrow \mathtt{list\ (V')}.$$

Jobs of mapping and reducing can be divided into smaller jobs, hence the calculations can be done in parallel by many nodes. A basic MapReduce flow is presented in Fig. 4.

The MapReduce framework is proprietary of Google Inc., but there are open source solutions offering similar approach, and among them, the best known and widely used is Apache Hadoop. From the Business Intelligence point of view map and reduce platforms like Hadoop can play a crucial role, as thanks to them complex processing of data, like aggregation or filtering, can be done in a very efficient

way. Business analysts can gain almost immediate access to tons of very accurate data without the need of using traditional, time-consuming ETL processes. In this sense such platforms (together with a database like HBase or any other NoSQL database that support Hadoop) can be an alternative or at least an important addition to traditional data warehouses used in today's Business Intelligence systems.

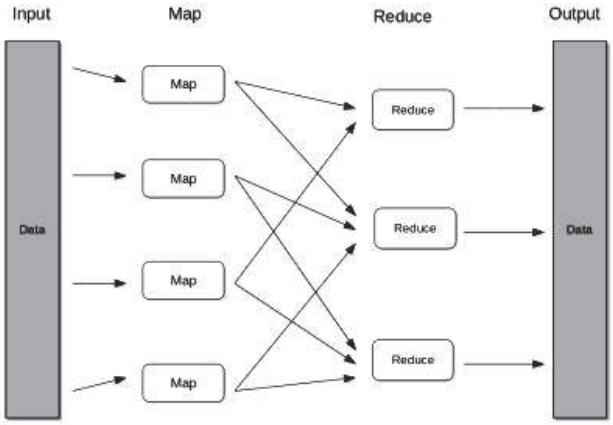

Figure 4. A basic flow in MapReduce data processing [8]

The example of such data warehouse is Apache Hive. Although it supports SQL-like commands (via HiveQL language) and stores its metadata in traditional RDBMS, the queries are translated into map and reduce jobs that are then executed on Hadoop platform.

3. Business Intelligence with NoSQL databases

Term Business Intelligence was first introduced in 1958 by IBM researcher H. P. Luhn [9]. One of the fathers of contemporary BI systems, H. Dresner from Gartner Research, in 1989 defined BI as "a broad category of software and solutions for gathering, consolidating, analysing and providing access to data in a way that lets enterprise users make better business decisions" [10]. However, from a technical point of view, a definition provided by E. Turban et al. can be seen as more precise: "An umbrella term that encompasses tools, architectures, databases, data warehouses, performance management, methodologies, and so forth, all of which are integrated into a unified software suite" [11]. The model of data delivery in a typical BI platform is shown in Fig. 5.

The core of each Business Intelligence system is a data warehouse (DW). During ETL (extract, transform and load) processes the data warehouse is loaded with data coming from ERP, CRM and other enterprise systems as well as databases

(usually RDBMS), and, if necessary, from any other source like spreadsheet files or flat files. Data in the warehouse are organized in a special form that will be presented later in the chapter. Once the data warehouse is loaded and updated, business analysts can perform some OLAP analysis, use data mining tools, or simply build reports tailored to their needs. ELT tools, data warehouse, analytical and reporting tools together constitute a BI platform (called often a BI suite).

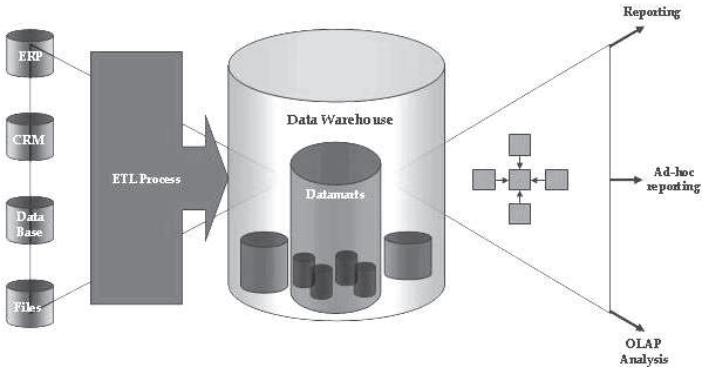

Figure 5. Typical BI architecture [12]

3.1. Data warehouses and OLAP servers architecture

Warehouses can use different structures, but typically they have at least three layers: staging, integration and access. Staging layer is responsible for storing a raw data extracted from source databases and flat files. In the integration layer data is integrated and transformed into an organized structure (data warehouse database), where data is usually arranged into dimensions and facts. Finally, the access layer is responsible for delivering data to end users.

The most widely used analysis in Business Intelligence is OLAP (on-line analytical processing). Typical OLAP operations include rollup (increasing the level of aggregation) and drill-down (decreasing the level of aggregation or increasing detail) along one or more dimension hierarchies, slice and dice (selection and projection), and pivot (re-orienting the multidimensional view of data) [13]. To perform OLAP operations quickly usually a special OLAP server is used either within the data warehouse or as a separate solution.

There are two main OLAP architectures: ROLAP (Relational OLAP) that stores analytical data in RDBMS and MOLAP (Multidimensional) that stores data in an optimized multi-dimensional arrays. Each architecture has some advantages over the another. ROLAP technology is generally better suited for models with very high credibility of dimensions (millions or more), and when no-aggregatable facts (like texts) are used. MOLAP generally performs better in typical OLAP ap-

plications than ROLAP [14], but it requires pre-processing (load from OLTP database or data warehouse) that can last for a long time. Load times in ROLAP are usually much shorter than in MOLAP and the server is more scalable, however, ROLAP is evidently slower especially with respect to aggregated data. To overcome this problem aggregations are stored in RDBMS as materialized views or as dedicated tables. This in turn requires a careful design of ETL process, because aggregate operations that were not previously cached can be a very time consuming. The advantages of both ROLAP and MOLAP models have been combined in HOLAP (Hybrid OLAP) model.

According to some business analysts current Business Intelligence platforms and data warehouses do not cover all the data necessary for decision making in today's complex economic environment. Big volumes of data (so called "big data") cannot be processed by traditional warehouses and OLAP servers that base on RDBMS solutions. Instead of them the solutions originated from the NoSQL movement like MapReduce (Hadoop), Hive, Pig or jaql should be applied.

3.2. Integration of NoSQL databases with BI platforms

Business Intelligence platforms supplied by the top vendors usually do not integrate with NoSQL databases, however, some of them (e.g. Oracle) provide a native access to Apache Hive. Instead of this, big vendors have recently started to provide their own NoSQL solutions (Windows Azure Storage, Oracle NoSQL) that in the future can probably be integrated with their existing BI platforms. Alternatively open source BI platforms can be used. The most popular ones are Jaspersoft and Pentaho. Both of them can natively read the data from the most popular NoSQL databases like Cassadra or MongoDB by their data integration (ETL) tools and reporting servers.

Also both platforms use Mondrian (officially called Pentaho Analysis Services Community Edition) as an OLAP server. A simplified architecture of Mondrian server is shown in Fig. 6.

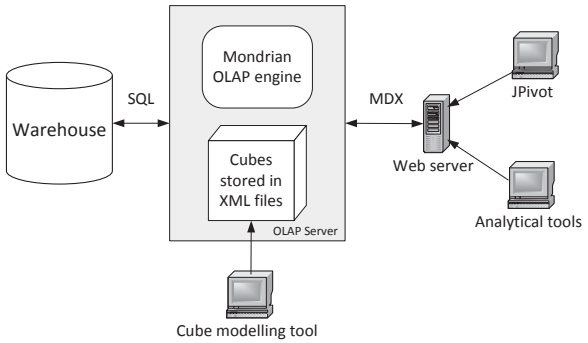

Figure 6. A simplified architecture of Mondrian server (based on [15])

Mondrian OLAP server stores cube schemas in XML files that can be modeled by client tools like Pentaho Schema Workbench and its engine is based on ROLAP model. This engine can communicate only with SQL language, so the data from NoSQL databases cannot be processed directly. First they need to be loaded into the data warehouse that is based on RDBMS like in traditional BI platforms. There are plans, expressed by some Mondrian community members, to give the users a possibly to replace SQL database in Mondrian with HBase or MongoDB.

Basically the same architecture as in Mondrian is used in much more lightweight solutions like e.g. an OLAP framework architecture written in Python (Cubes) [14]. The architecture of such a lightweight server is shown in Fig. 7.

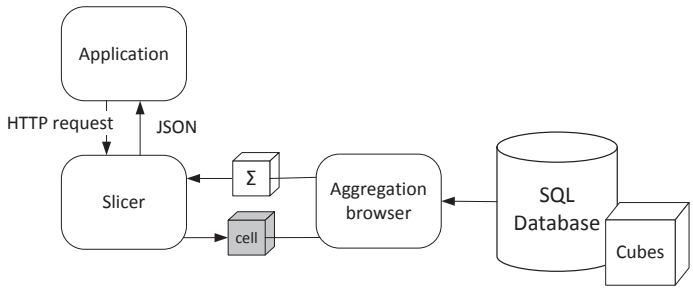

Figure 7. A simplified architecture a lightweight OLAP framework (based on [16])

A client sends a request for some cell of the cube to the server (called in this case a slicer), the server uses appropriate aggregation browser backend to compute this request and a result in JSON format is sent back to the client. Although many different aggregation browsers have been originally planned, including a MongoDB browser, it has not been implemented yet. The only aggregation browser is currently "SQL denormalized" that stores the data from converted star schema in the form of a deformalized views and tables.

3.3. New OLAP model using MapReduce framework

Big data stored in NoSQL databases can be processed during ETL processes using map and reduce functions defined within Hadoop platform (both Pentaho and Jaspersoft BI platforms have such ability). Typical scenario for big data analysis is currently to use e.g. Hive in Hadoop environment (for OLTP) and a ROLAP tool for data analysis. Sometimes such solution may occur not enough. If more a lot of data is added to the Hive warehouse in almost a real time, the aggregated data stored in RDBMS for OLAP purposes may quickly become obsolete or at least inaccurate. Moreover, when really huge data sets are used, the queries performed on partially aggregated data can be too slow for OLAP browsing.

Thus, the idea is to use MapReduce paradigm not only on OLTP side, but also for OLAP processing. Map and reduce functions can be programmed relatively easy using HQL provided by Hive or Pig Latin language.

Also recently some new projects focusing strictly on OLAP processing like e.g. olap4cloud [16] has emerged. Olap4cloud is based on HBase and Hadoop, but besides MapReduce it uses data defragmentation, intensive indexing and preaggregations. The developers of olap4cloud performed the test in which aggregation query was executed with 300 million rows of randomly generated 3 dimensions with values in [0..10], [0..100] and [0..10000] ranges and 3 measures with values in [0..100], [0..1000] and [0..10000] [16]. The goal of the query was to sum few facts across one dimension and filtering them on the basis of other two dimensions. The query that was used in tests is shown in Fig. 8.

```
select d3, sum(m1), sum(m2), sum(m3) from facts
where d1 = 1 and d2 = 1 group by d3
```

Figure 8. Test query for Olap4cloud performance experiments [17]

The test was run on few machines with average speed of processor 2 GHz and average RAM size of 1.7 GB. Unfortunately the number of nodes has not been specified by the authors. Nevertheless the obtained results are interesting. For Hive platform query executed in 23 minutes, while for olap4cloud it took only 26 seconds, when no pre-aggregations had been used, and impressive 5 seconds with pre-aggregations [17].

To compare those results with a traditional RDBMS the author imported the same 300 million rows into MS SQL 2008 Server that was run on a single machine with 8GB RAM and 2.0GHz quad core processor. Although the import process itself took for 30 minutes (the import was done by SQL Server Integration Services – ETL tool for MS SQL Server) the query executed in 135 seconds for the first run and only 12 seconds for the subsequent runs (SQL Server cached the data). This can simulate a simple ROLAP.

Additionally the author has also tested a new aggregation mechanism, that does not require the map and reduce functions written in an explicit way, and that was introduced in MongoDB 2.2. Fig. 9 presents the code for MongoDB expression that is equivalent to the query presented in Fig. 8.

```
db.bigdata.group({key:{d3:true}, cond: { d1:1,d2:1 }, reduce:
function(obj,prev) { prev.m1sum += obj.m1;prev.m2sum += obj.m2;
prev.m3sum += obj.m3;},initial: { m1sum: 0, m2sum: 0, m3sum: 0}});
```

Figure 9. Test query written as MongoDB expression

The query initially took 10 minutes to complete, but after indexes for “d1” and “d2” were created the query time was reduced to only 23 seconds. In the latter

case it was only two time slower than for MS SQL server. It can be farther improved by using MongoDB port for Hadoop to speed up map and reduce functions that are then performed on many nodes.

4. Conclusions

Experimental projects with non-relational databases developed at Google, Facebook, Yahoo or Adobe slowly begin to reach their maturity and now are used not only in garage projects, but are seen as a significant competition for the traditional RDBMS databases. Huge amounts of data are no longer restricted to meteorological, geographical, astronomical, biological or physical applications, but can be found also in today's business. A lot of sensors, logs, media devices, tracking systems generate even petabytes of data, so called big data (e.g. Google processes about 24 petabytes of data per day). Such data should be analyzed in tools corresponding to the BI tools used for traditional transactional data stored in RDMBS.

Currently most of the BI platforms and other OLAP-related tools that are capable to deal with huge NoSQL databases reproduce the well-known ROLAP model existing in traditional BI platforms. Although the MapReduce paradigm can be used, but only on OLTP side, while OLAP side is still dominated by RDBMS. There are two main reasons for that. First is the SQL language and well-known optimization techniques such as materialized views, which proved their efficiency for OLAP purposes. The second, even more important reason, is that traditional models can be easily applied to both non-relational and relational environments. In fact, we are at the stage where traditional warehouses are supplemented with big data platforms (like Hive) and coexist in one universal BI platform. However, sometimes this may be not enough for business needs and traditional approach for OLAP should be replaced with a new one, based on techniques like MapReduce. The results presented in some experiments are quite optimistic and lead us to believe that new approach for BI platforms can be put into practice in near future.

REFERENCES

[1] *NoSQL A Relational Database Management System*, http://www.strozzi.it/cgi-bin/CSA/tw7/I/en_US/NoSQL/Home Page, Retrieved 6 September 2012.

[2] Lith A., Mattsson J. (2010) *Investigating storage solutions for large data - A comparison of well performing and scalable data storage solutions for real time extraction and batch insertion of data*, Department of Computer Science and Engineering, Chalmers University of Technology, Göteborg.

[3] http://nosql-database.org, Retrieved 6 September 2012.

[4] Abadi D. J., Turner M. J., Hammond R., Cotton P. (1979) *A DBMS for large statistical databases*, VLDB '79 Proceedings of the 5-th international conference on Very Large Data Bases.

[5] Abadi D.J., Boncz P.A., Harizopoulos S. (2009) *Column oriented Database Systems*, PVLDB 2(2), 1664-1665.

[6] Bajaj P., Dhindsa S.K. (2012) *A Comparative Study of Database Systems*, International Journal of Engineering and Innovative Technology, Volume 1, Issue 6

[7] Dean J., Ghemawat S. (2008) *MapReduce: Simplified Data Processing on Large Clusters*, Communications of the ACM, Volume 51 Issue 1, 107-113.

[8] Duarte de Souza R.G. (2010) *MapReduce "Easy distributed computing",* http://girlincomputerscience.blogspot.com/2010/12/mapreduce.html, Retrieved 8 September 2012.

[9] Luhn H.P. (1958) *A Business Intelligence System*, IBM Journal 2 (4), 314-319.

[10] Chee T., Chan L., Chuah M., Tan Ch., Wong S., Yeoh W. (2009) *Business Intelligence Systems: State-Of-The-Art Review And Contemporary Applications*, Symposium on Progress in Information & Communication Technology 2009.

[11] Turban E., Sharda R., Delen D., King D. (2010) *Business Intelligence*, 2nd edition, Prentice Hall.

[12] http://etl-tools.info/en/bi/datawarehouse_concepts.htm, Retrieved 10 September 2012.

[13] Chaudhuri S., Dayal U. (1997) *An Overview of Data Warehousing and OLAP Technology*, SIGMOD Record 26(1), 65-74.

[14] Bach Pedersen T., Jensen C. (2001) *Multidimensional Database Technology*, Distributed Systems Online (IEEE), 40-46.

[15] *Cubes OLAP avec Mondrian*, http://www.osbi.fr, Retrieved 11 September 2012.

[16] *Cubes - OLAP Framework*, http://packages.python.org/cubes, Retrieved 11-09-2012.

[17] *olap4cloud. User Guide*, http://code.google.com/p/olap4cloud/wiki/UserGuide, Retrieved 12 September 2012.