Fuzzy Photo Data Warehouse: Database Model Requirements

Jethro Shell

December 3, 2012

1 Introduction

Within this document there will be two main areas discussed. The first will be an overview of the software considerations, primarily the Operating System (OS), when implementing a database model. Based on these considerations, a proposal will be made in regard to the type of a software operating platform to be used. Secondly a discussion will be made in respect to the implementation of a database model in regard to a data warehouse. An overview will be given of the requirements of the data warehouse for the Fuzzy Photo Project, which will be followed by an analysis of possible applicable database structures. This section will culminate with a suggestion for a database model.

2 Platform Considerations

There are a number of considerations that need to be taken into account when approaching the construction of a database system in terms of platform base. Five broad areas drive the decision making process:

1. **Performance** - Key parameters used to ascertain the level of throughput such as number of requests or transactions associated with a database operation.
2. **Security** - Configuration, maintenance and support of the security structure of the OS.
3. **Maintenance** - The ease and cost associated with the maintenance of the database structure in the context of the Fuzzy Photo project.
4. **Cost** - Initial outlay and continued support costs.
5. **Compatibility / Legacy** - Ease of use and compatibility with the structures already in place associated with the incoming system.

Based upon these areas, three operating system platforms (Apple, Unix/Linux and Microsoft) will be compared and assessed in order to understand their applicability.

2.1 Apple Operating Systems

Apple has produced a number of products that now support server based applications. The Apple Mac OS X server is installed on a number of proprietary systems. With a high ease of use, Mac OS X server brings a number of features that allow the user to easily access and maintain a system. The latest OS provides a large feature set containing generalised and dedicated features such as *Time Machine*, a system to centralise the back up of the server. On top of the ease of use, the Unix BSD base of the operating system (See table 1 for further details) allows the incorporation of some functionality that exists within Unix-compliant Operating Systems (OS's).

2.2 Unix / Linux Operating Systems

There are a large number of distributions based on Unix, and as an off shoot of the Unix kernel, the Linux operating system. Some of the key OS's currently used within server applications will be summarised within this section.

Table 1 shows a summary of the key OS's that are currently distributed, highlighted some of the main attributes. An over riding feature of Unix and Linux based systems are the depth of availability and the open source nature of the software. Predominantly the systems are free to acquire, install and operate with options to receive *paid for* support. The open source nature of the software distribution has produced a large community of support that can act as a structure additionally or as an alternative to commercial support.

Both Unix and Linux based OS's can support the use of structured database models such as relational databases in the form of query languages (SQL). The Linux Apache MySQL PHP/Perl/Python (LAMP) installation forms the bedrock of a multitude of web servers.

Distribution	**Base Distribution**	**Free / Non-Free**	**Updates / Support**	**Description / Information**
Red Hat Server	Red Hat Linux	Non-Free	Supported for 2-3 Years. Paid for support.	Red Hat is supported by Red Hat Inc. It is a popular OS for large enterprise systems. It is stable and reliable as any issues are rectified by developers. For a free alternative there are CentOS and Fedora.
Suse Linux Enterprise Server	Suse Linux, Novell	Non-Free	Major version 3-4 years, support pack 18 months.	A mature, well tested OS that releases versions at long increments to prevent issues arising.
OpenSuse Server	Suse Linux, Novell	Free	Updates every 8 Months approx.	Free version of the Suse project. Developed as an off shoot. It is supported by a community of developers.
Debian Server	Debian	Free	2 years for each release with incremental updates.	Highly configurable, lightweight OS that forms the basis of many other systems.
Ubuntu Server	Debian	Free	Long Term Support (LTS) version supported for 5 years. Canonical can offer paid for support.	Popular OS with a large community of support. Release dates are more frequent than other OS'ses which can bring advantages and drawbacks.
CentOS	Red Hat Linux	Free	Released every 3 years with support from 3-5 years.	A well maintained free fork of the Red Hat OS. Release tracks the Red Hat development so in turn patches and bugs fixed by Red Hat developers filter through.
FreeBSD	FreeBSD (Unix-compliant)	Free	2 years support for each release.	There are many derivatives of the FreeBSD system (see previous section regarding Apple). The system is built for heavy traffic use (Microsoft and Yahoo are known to use it), however support can be restricted.

Table 1: Unix/Linux Distributions

2.3 Microsoft Windows

At the time of writing, the current version that is offered by Microsoft is Windows Server 2012, a mature OS used in many large data centres across corporate organisations. The Windows Server series has a large body of support and is a well maintained product. Available in a multitude of flavours, Microsoft are able to offer systems from home servers to large corporate installations.

As with both the use of Unix/Linux systems and Mac servers, Windows can offer a multitude of software applications to support web based services. In parallel to Mac Apache MySQL PHP/Perl/Python (MAMP) and LAMP, a Windows Apache MySQL PHP/Perl/Python (WAMP) installation is available.

The economic costs associated with the use of a Microsoft based OS are dependent on the variety used and the support system chosen. The proliferation of Windows based OS's also requires more attention to be paid to the question of security.

2.4 Comparisons

2.4.1 Performance Comparisons

To understand the performance of varying operating systems whilst undertaking distinct database processes, performance metrics are required. The Transaction Processing Performance Council (TPC) provides processing and database benchmarks for industry bodies. A number of differing metrics are used across varying database and hardware configurations. A high proportion of the results indicate that variants of the Linux OS perform well. A TPC benchmark for ad-hoc, decision support queries, TPC-H shows high performance from Linux based systems. TPC-H is a suite of business orientated queries and data modifications. The benchmark has been designed to examine large volumes of data and execute queries with a high degree of complexity.

As of the 8th November 2012, the top ranked result in the 100GB group emerged from a cluster based Linux system. The EXASOL EXACluster OS 4.0 based on a Dell PowerEdge R710 achieved 1,112,401 Query-per-Hour (QphH). The second highest mark resulted from a Dell PowerEdge R720 using Red Hat Enterprise Linux 6.1 producing 403,230 QphH. A comparable Microsoft system produced 73,974 QphH using a HP ProLiant DL380 G7 [13].

By removing all clustered systems, Unix/Linux operating systems continue to be the top ranked OS's across the database range. The Mac OS X does not feature in this list.

2.4.2 Security Comparison

It is difficult to make an overarching comparison of the systems *out of the box* as there are differing processes that need to be taken into account in each case.

Due to the extensive nature of Windows and the longevity of Unix combined with its evolution into Linux, there have been a greater quantity of comparisons made between these OS's. In a report commissioned in 2005 by the Robert Frances Group sponsored by IBM, they reported that key enterprise buyers found that it was easier to lock down a Linux system then a comparable Windows OS. Additionally the deployment of patches produced minimal downtime [6]. Microsoft systems have made significant advances in recent years in security related additions to the systems alongside the availability of numerous security applications. Equally Apple have a number of available applications and in built systems to secure servers.

2.4.3 Maintenance Comparisons

There are multiple considerations to take into account when considering the maintenance of a server, many of which can be driven by previous knowledge and the server requirements. Most modern server software implementations supply applications to assist in the maintenance of the underlying server structure. Microsoft Server 2012 offers a Maintenance Plan Wizard that assists the server user in constructing database administration tasks. The wizard allows for easy construction of processes such as backups, integrity checks, and statistic updates.

Apple Mac OS X Server offers a simple, accessible way to maintain a server. The latest incarnation of the server structure however is less well suited to an enterprise application. A number of inherent applications have been removed or are less well supported. There has been a move toward the requirements of the use of the command line (an approach similar to some Linux distributions) or third-party applications.

Unix / Linux distributions are renowned for their resilience to failure. Many distributions are at the heart of large data warehouses and data marts (large companies such as Google, IBM, Panasonic and Wikipedia are adopters of Linux for their server systems [1]). Linux especially has progressed significantly in the last 20 years in terms of ease of use and maintenance structures.

2.4.4 Total Cost of Ownership Comparisons

Both Windows and Linux can be deployed on a variety of hardware architectures. The proprietary nature of the Apple OS X OS results in a stricter deployment. Apple products have greater restrictions on the hardware that the OS can be implemented upon. This reduces the options with which vendors can offer server packages. This can have a large impact on the overall decision making process (see Sections 3.1.1).

The variants of the Unix/Linux OS have a financial incentive as they predominantly offer free distributions. Varying levels of support can be found for differing distributions. Eminent distributions such as Ubuntu have strong community and *paid for* support structures.

The cost associated with the installation of the Windows and the Apple server OS varies largely. The current Apple server has a low cost as it is largely seen to be focussed at the small business user. As an add on to the current OS, Apple charge a nominal fee [1].

As with other Windows software packages, a number of server variants are offered to the consumer. Based on the requirements of the application, different licenses can be purchased and so in turn different costs applied. The current version of Microsoft Server essentials (the lowest end version) is shipped with server hardware as standard, with higher end versions costing many thousands of pounds.

2.4.5 Compatibility / Legacy with Supporting Systems

Currently the Fuzzy Photo project houses systems that use OS's from both Apple and Linux. An implementation of a system based on either of these systems can assist in the ease of any required transition. The movement between fewer operating systems reduces support costs and improves system support overall. Similarity between OS's provides a basis with which to migrate the support structure. As Linux systems are

[1]At the time of writing the cost of adding OS X Server was £13.99

managed similarly across multiple sub-platforms, they allow for skills to be distributed across them.

Additionally, the iterative production of Apple and Microsoft products also allows for skills previously acquired to be transferred from one product to another. Similarly, the underlying Unix layers of both the Apple OS X and Linux OS's allows for greater integration of these two systems.

2.5 Suggested Operating System Strategy for the Data Warehouse Implementation

A number of varying OS strategies can be used to implement the data warehouse. Taking into account the discussion offered in the previous sections, a Unix/Linux system is considered the best option. Below is a summary of items taken into consideration:

Performance Although heavily dependent on the database software, a considerable quantity of high transactional systems use a Unix/Linux based system to support the underlying structure.

Cost implications A linux distribution can be sourced for free, against a cost associated with other systems.

Support / Maintenance Major Linux distributions have both a large community base of support and the possibility of licensed support structures. Many distributions are extremely mature and are implemented in substantial commercial operations [15]. As a result issues surrounding maintenance are minimal.

Legacy The Unix nature of the OS X underlying system and the Linux OS used within other systems within the Fuzzy Photo project allow for the transfer of skills and the continued support of legacy systems within the new implementation.

3 Implementation of a Database Model

Database and data model are often referred to as the same concept. Within this document, this terminology will be used interchangeably. A database model can be described as a collection of conceptual tools for representing real-world entities. These objects are modelled and the relationships among them. Data models can differ in the way in which the available data can describe them and the amount of semantic detail that can be expressed within them [11].

The next sections will provide an overview of the requirements of the data warehouse specifically focussing on the Fuzzy Photo Project. This will be followed by a discussion presenting possible data model solutions. Finally a suggested modelling approach will be given.

3.1 Fuzzy Photo Data Warehouse Requirements

Form and function of a data warehouse can be varied though it is considered to be a key component of the management of decision making [10]. Fundamentally a data warehouse extracts selected information in advance, translates, filters and merges the information so that it can be stored in a repository. Queries posed to the system are evaluated directly without accessing the original information sources [12].

Taking these basic concepts, Figure 1 shows an overall view of the Fuzzy Photo data warehouse in respect to the companion data sources.

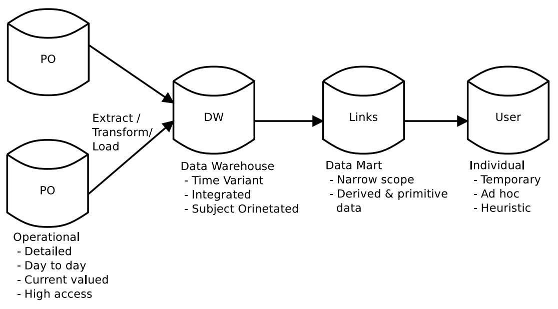

Figure 1: Overall Database Structure.

Here it can be seen that each of the database structures has differing data specifications. The Operational database (depicted as being housed within the Partner Organisations (PO's)) is a day to day system that is highly accessed. The data warehouse (DW) integrates this data through an extract/ transform/ load process. The data mart (which within the Fuzzy Photo Project can act as the links database) has a narrow, specified scope using both derived and primitive data. The final user interaction houses temporary and ad hoc information based on individual requests.

Based on this view, the requirements for the Fuzzy Photo data warehouse can be summarised below:

Storage It is envisaged that the data warehouse will house very large data sources.

Scalability As the data increases in size, there will be a requirement for the database to scale with no effect on the underlying performance.

Reliability There is a need for a reliable execution of transactions with significant processes to support failure.

Complexity Highly complex (for example, numerous operations) need to be accommodated within the database structure.

3.1.1 Hardware Considerations for Data Warehouse

Patterns of Hardware Usage There are differences in the patterns of hardware usage between a data warehouse and an operational server. Figure 2 shows the classic pattern of usage for both server types.

On the left is the Operational Server. There are peaks and valleys, but overall it is relatively static and predictable. The data warehouse environment is far more reminiscent of binary behaviour. To try to optimise a server for both types of behaviour, as a result, is extremely difficult. The focus of the hardware implementation of the Fuzzy Photo data warehouse will be upon the optimisation of the server towards this structure. In the following sections, the requirements of the data warehouse will be discussed (both general and specific).

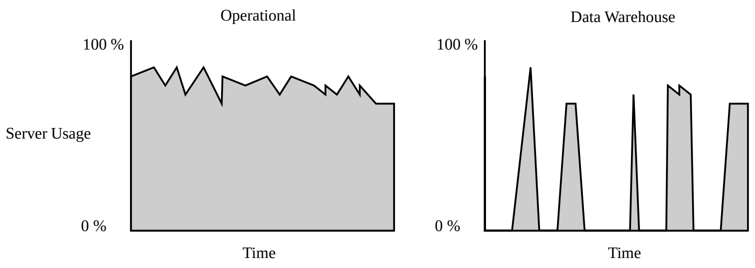

Figure 2: Example of the Variations in the Hardware Usage for Operational and Data Warehouse Servers [8].

Availability and Persistence A key requirement of the information held within data warehouses is the availability of the data. Although not executing high transaction rates, there is a need to have the information available. This results in storage options becoming a key consideration. High volumes of data require the use of larger quantities of disks. Increasing the number of disks has a direct relationship to the Mean Time To Failure (MTTF) of the storage. It is envisaged that the implementation of the Fuzzy Photo data warehouse will hold a significant quantity of data that will require a multiple disk infrastructure.

Activity Data warehouse transactions are generally based exclusively on read activity, with a limited number of writes often carried out in batches. The operations can occur in large bulk tasks where large quantities of data are read from the database all at once in order to respond to a reporting or analytic query. The Fuzzy Photo data warehouse will adopt this type of activity. The most common bottleneck imposed on a data warehouse is the Input/Output structure.

Restore It is a general requirement of data warehouse technology that it operates a fast restore feature so that partial or full tables can be returned. This is a requirement of the Fuzzy Photo data warehouse. There are options to either restore data from secondary storage or alternatively from Direct Access Storage Device (DASD). The standard practice is to double the amount of DASD to accommodate recovery. The incorporation of automated data corruption detection is also highly recommended.

3.2 Relational Model Implementations

The application of a structured format such as a relational model requires a number of attributes to be applied to the data. It is necessary for the data to be organised in semantic chunks. This forms entities within the database. The production of entities can be amalgamated into similar groups that are defined as relations or classes. The entities that are in the same group can have the same descriptions otherwise known as attributes. The descriptions for all entities in a group can be defined as a schema. There are benefits and issues associated with such a structure. The following sections highlights the advantages and limitations of the use of relational databases.

3.2.1 Advantages of Relational Databases

Precision Across relational database implementations there is support for Atomicity, Consistency, Isolation, Durability (ACID) transactions. ACID is a set of constraints and is defined as: **Atomicity** states that all database modifications must follow an *All or Nothing* approach. **Consistency** states that only valid data will be written to the database. **Isolation** requires that if multiple transactions occur, they do not impact each other. Finally, **durability** ensures that any transaction committed to the database will not be lost.

Feature Set Relational databases are very mature and are able to offer a large feature set.

SQL Relational databases are synonymous with the use of SQL. SQL is convenient with structured data. It is designed to work with a structured approach using relational, organised databases with fixed table information [9].

Data Relational databases are able to focus on small amounts of data on a single server using tables housing different types of data formats. Results of any query can be simple or complex with multiple search conditions.

3.2.2 Limitations of Relational Databases

Complexity Users must convert data to a table form in order to use relational databases. If the data does not work in this form, then difficult, complex structures can arise.

Limited Capacity Standard relational database have difficulty in supporting big data structures.

Scalability Issues The use of joins across tables impedes the use of scalability in standard relational database structures [7]. Scalability can be achieved with more powerful and so more expensive hardware though only to a certain point.

The size of a single implementation of a relational database can be large but as with many implementations that are placed onto a single unit, this can be limited by accessibility to the data via memory. Estimates have shown that the maximum size a single MySQL table could be 256 terabytes [5].

Some of the limitations that are encountered in the implementation of a relational database can be overcome through the use of a clustered structure. Database developers have implemented systems that use clustering to negate scalability issues as well as using in memory architectures. MySQL's Cluster software has implemented these measures. The application shards data over multiple database servers as opposed to a single centralised database. Each shard is replicated allowing scalability horizontally. VoltDB uses similar measures. Tables can be partitioned over multiple servers or selected tables can be replicated. The database is designed to fit within distributed RAM on the servers so disk waits are limited [4].

3.3 Analytical Model Implementations

Analytical database models increased in popularity alongside the growth in volumes of data. Commonly described as Online Analytical Processing (OLAP), this form of implementation allows for the extracting of data from various data repositories to construct a compatible source. There are two main architectures used within OLAP distributions: Multi-dimensional Online Analytical Processing (MOLAP) and Relational Online Analytical Processing (ROLAP).

MOLAP This form uses a multi-dimensional database to provide analysis using the Multi-Dimensional eXpressions (MDX) language. Generally most of the data is required to be pre-compiled to provide acceptable query performance. As a result it performs best when the data set is relatively small (fewer than 5gb) [2].

ROLAP The architecture in a ROLAP database is provided by a relational structure within the database. A ROLAP database is very efficient at supporting dynamic and volatile data sources. Equally it is able to scale to large volumes of input data with large dimensions.

The next two sections discuss the advantages and disadvantages of an OLAP approach in a broader sense.

3.3.1 Advantages of Analytical Databases

Speed The data can be organised for rapid querying and analysis through the use of pre-aggregation and pre-compiling.

Viewing Various tools allow for easy viewing of information by developers and users.

Dimensionality The structure of the database allows for the analysis of data across many dimensions, each dimension summarising an attribute of the system the database represents.

3.3.2 Disadvantages of Analytical Databases

Size MOLAP implementations can have size limitations. As the number of dimensions increases, the number of cells can increase exponentially. As a result the maximum cell count can be easily reached. For example, a 16 dimension database with 5 members in each has 152 billion cells.

Access Within MOLAP the speed of access to data can be limited to the precompiled cubes. Also within ROLAP forms, speed of access is bound by the SQL structure of the database as with relational databases.

Knowledge The multi-dimensional cube structure of MOLAP databases are a technology that is not often within an organisation so needs to be learnt. As a result the database can take longer to implement and be more difficult to use.

LucidDB represents an implementation of an analytical database model. LucidDB is very fast at bulk-loading or updating large amounts of data at once, but it is not intended to work well for the single-row operations typical of transactional systems. It can be used as a data warehouse, data mart, or operational data store in tandem with the traditional transactional systems used as data sources [3]. Often OLAP systems are used within the application layer representing the data mart. This may be a consideration for the links database system as an OLAP database can sit on top of a data warehouse.

3.4 NoSQL Database Implementation

In recent years there has been an increase in the development of database structures that are commonly referred to as *NoSQL*. Although indicating the non-use of the SQL language, it in fact references *Not Only SQL* or *Not Relational*. The overall structure of this type of database spans a number of differing implementations, and the feature set is not entirely agreed upon. Cattell [4] has defined six possible key features:

1. Ability to horizontally scale simple operation throughput over many servers.
2. Ability to replicate and distribute data over many servers.
3. Simple call level interface or protocol.
4. Weaker concurrency model than ACID.
5. Efficient use of distributed indexes and RAM for storage.
6. Ability to dynamically add new attributes to data records.

Within the hierarchy of NoSQL databases, there are differing types and implementations that use different data stores. These can broadly be summarised as: Key Value, Big Table, Document and Graph Databases.

Key Value Stores A key store is the simplest form, using a single key-value as the index for all data provided by the programmer. These databases generally provide a persistence storage mechanism alongside additional functionality.

Column / Big Table Implementations Column based databases contain one extended column of closely related data. This is in contrast to relational databases that use a standardised and heavily structured column and row. Apache Cassandra is a Big Table variant that is able to handle very large data by spreading it across commodity servers. It was initially developed to support Facebook.

Document Stores These systems store documents as the user has defined. The documents are indexed and a simple query mechanism is provided [4]. Users can add any number of fields of any length to a document making the structure very flexible.

Graph Databases A graph database takes the form of a graph structure with nodes, edges and properties to store data. Each element within the structure contains a pointer to the adjacent node which allows for a lack of index lookups.

As has been illustrated, there are many NoSQL implementations with a broad selection of features. In the following sections an overview will be given of the advantages and disadvantages of using a NoSQL based system.

3.4.1 Advantages of NoSQL Databases

Speed Databases that use the relational form are bound to ACID constraints. Performing to these constraints on every piece of data makes them slower. NoSQL databases conform to a more relaxed set of constraints, the Basically Available, Soft State, Eventually consistent (BASE) structure. As a result they are quicker to execute many forms of transaction [14].

Scalability NoSQL databases allow for *Shared Nothing* horizontal scaling producing replication and partitioning of data. This in turn allows for the support of a large number of simple read / write operations per second.

Volatile Data There is no need for a strict schema format within NoSQL databases which lends itself to the use of volatile data, unlike relational databases.

3.4.2 Limitations of NoSQL Databases

Overhead The need to use manual query languages in NoSQL databases can produce time consuming activities when complex queries are needed.

Reliability NoSQL databases do not natively support ACID. Additional programming is required to achieve this goal.

Unfamiliarity There are many variations to the NoSQL database structure and the technology is relatively new. Most organisations do not have experience of using the technology which can lead to issues and impede development [9].

3.5 Suggested Database Implementation for Data Warehouse

Taking into account the discussion presented in the previous sections, the use of a relational database is suggested. It is suggested that the database take a standardised form initially with a move toward a clustered, scalable approach as the project progresses. The initial size of the implementation is currently yet to be ascertained, however it is known that there is a requirement for the database to be scalable and resilient to failure. The use of a cluster structure will allow for this. It is noted that scalable relational databases offer good per-node performance as well as scalability, possibly comparable to NoSQL databases [4]. This is combined with the ability to offer the convenience of the SQL language and ACID properties. The structure of the database is strict taking the form of standard schema which also is applicable to the use of a relational database form.

A number of implementations can be used, including MySQL Cluster which currently offers an open source version of the database software.

References

[1] 50 places linux is running that you might not expect. http://www.comparebusinessproducts.com/fyi/50-places-linux-running-you-might-not-expect, March. Online: Accessed on 3rd December 2012 Updated: March 23, 2010.

[2] The case for relational olap: A white paper prepared by microstrategy, incorporated. http://www.cs.bgu.ac.il/ onap052/uploads/Seminar/Relationalrategy.pdf, November 2012. Online: Accessed on 30th November 2012.

[3] Luciddb architecture. http://www.luciddb.org/html/arch.html, November 2012. Online: Accessed on 28th November 2012.

[4] R. Cattell. Scalable sql and nosql data stores. *ACM SIGMOD Record*, 39(4):12–27, 2011.

[5] Nick Duncan. Maximum mysql database size? http://nickduncan.co.za/maximum-mysql-database-size/, November 2009. Online: Accessed on 30th November 2012.

[6] Robert Frances Group. Tco for application servers: Comparing linux with windows and solaris. Technical report, Robert Frances Group, August 2005.

[7] Jing Han, E. Haihong, Guan Le, and Jian Du. Survey on nosql database. In *Pervasive Computing and Applications (ICPCA), 2011 6th International Conference on*, pages 363 –366, oct. 2011.

[8] W.H. Inmon, W.H. Inmon, W.H. Inmon, and W.H. Inmon. *Building the data warehouse*. J. Wiley, 2002.

[9] N. Leavitt. Will nosql databases live up to their promise? *Computer*, 43(2):12 –14, feb. 2010.

[10] B. Shin. An exploratory investigation of system success factors in data warehousing. *Journal of the Association for Information Systems*, 4(1):6, 2003.

[11] A. Silberschatz, H.F. Korth, and S. Sudarshan. Data models. *ACM Computing Surveys (CSUR)*, 28(1):105–108, 1996.

[12] D. Theodoratos, T. Sellis, et al. Data warehouse configuration. In *Proceedings of the International Conference on Very Large Data Bases*, pages 126–135. INSTITUTE OF ELECTRICAL & ELECTRONICS ENGINEERS (IEEE), 1997.

[13] TPC. Tpc-h - top ten performance results - non-clustered version 2 results. http://www.tpc.org/tpch/, November 2012. Online: accessed 8th November 2012.

[14] B.G. Tudorica and C. Bucur. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1 –5, june 2011.

[15] Steven J. Vaughan-Nichols. Amazon ec2 cloud is made up of almost half-a-million linux servers. http://www.zdnet.com/blog/open-source/amazon-ec2-cloud-is-made-up-of-almost-half-a-million-linux-servers/10620, March 2012. Online: Accessed on 28th November 2012.