NoSQL Databases

Silvan Weber
Master of Science in Engineering
University of Applied Sciences HTW Chur, Switzerland
www.htwchur.ch/mse
silvan.weber@msc.htwchur.ch

Abstract

Conventional SQL databases use SQL (structured query language) as their primary interface to manage databases and are based on a relational database model. The term "NoSQL databases" is an emerging general expression for databases with the aim to not (or almost not) use the functionalities of SQL databases. NoSQL databases shall be non-relational, distributed, open-source and horizontally scalable as the NoSQL archive states. This article clarifies the meaning of this statement, points out the problems of SQL as well as their possible solutions provided by NoSQL and shows the main applications and categories of NoSQL databases.

1 Introduction

In this paper, we will use SQL databases, RDBMS (relational database management system) and relational databases as synonyms, despite the fact that the two terms do not necessarily have the same meaning all over the world of databases.

1.1 History

The whole movement against the monopoly of SQL databases began in the late 1990s. However, it wasn't before 2009 that it became a serious competitor to RDBMS. There was no clear definition or term for this movement, until the expression NoSQL - meaning "not only SQL" - was created, suggesting the *avoidance of SQL*. It was then that NoSQL became very popular thanks to the existing and very well-known term SQL itself.

1.2 Definition

The NoSQL archive defines NoSQL databases as "non-relational, distributed, open-source and horizontal scalable" [Arch10].

- Non-relational means that the database is not based on a relational database model anymore and thus has no relations (tables) - the principle is completely different.
- Distributed means that the data is stored on and managed by different machines, so there could be data replication.
- Open source means that everyone can look into the source code for free, change it and compile it themselves. However, this term should not be taken too seriously. It is just meant as an opposite of commercial RDBMSs, which sometimes do not quite fit the requirements despite the fact that they cost a lot of money according to [Edli10].
- Horizontally scalable means that the more database servers you add, the more performance you get in a nearly linear way. This is also called out scaling. SQL databases do not naturally scale horizontally because the latter isn't a core business of relational DBs.

According to [Edli10], the definition of the NoSQL database could be extended with other points. They adopt the same four criteria but add that a NoSQL system is schema free or has weak schema restrictions, shared-nothing (the nodes are independently of each other), that there is a simple data replication, the system provides a simple API (application programming interface) and that the consistency model is not ACID anymore. However, [Edli10] also states that NoSQL databases *most often* use *some* of these points and not necessarily all of them.

Usually NoSQL databases do not use JOINs (i.e. links between tables) or constraints (like NOT NULL) anymore. Also NoSQL databases are not intended to handle complex queries over multiple tables. The best architecture for such queries is still the relational model.

1.3 Distinction of other terms

1.3.1 Non SQL

The term "Non SQL databases" is misleading, not to say wrong. It predicates that these databases are without any use of SQL and this is not always the case in NoSQL. The term "Non SQL databases" does exist, however it is rather used as a vague term than as a professional expression. It means that the data is treated by another language than SQL, e.g. XQuery for XML databases [Disk10]. Some NoSQL databases are non SQL databases.

1.3.2 Distributed storage

Related to NoSQL there are terms such as "distributed storage" or "distributed structured storage". It is quite difficult do distinguish such terms from NoSQL, because they exactly describe one feature of NoSQL.

Distributed storage is a generic term for a system which pretends to be a single storage but in reality is a collection of many computing units storing parts of the files [Dist08]. Some NoSQL databases claim to be a distributed storage system, e.g. Google's BigTable [BigT06].

2 Fundamentals

2.1 ACID

In the SQL environment there is the ACID concept (atomicity, consistency, isolation, durability). This paper will not go into the ACID concept, only into the problems it could make for NoSQL.

As you can imagine, implementing ACID into distributed systems is not trivial. The major problem is the consistency of datasets being distributed over several servers. If one tuple in one table changes, every user of this tuple has to be informed immediately so that the consistency is always given.

2.2 BASE

BASE (Basically available, soft state, eventual consistency) is not just a chemistry hoax to ACID. It is more optimistic regarding the consistency of replications. It is intended that the *consistency* after a transaction is *not a solid state* anymore (soft state). It shall be reached not right after finishing the transaction, but rather *sometime* (eventually consistent). The main focus of BASE is the *permanent availability*. BASE is the opposite of ACID. NoSQL databases are classified in-between the spectrum from ACID to BASE.

In the case of a bank, the eventual consistency is not what you want, thinking about two different balances on different servers! The balance must be equal just in time in every database involved in a money transaction session. In the case of an online book trade, the "just-in-time consistency" becomes less important. It does not matter if a book's price on one replication differs from another during a short time like a few hours.

2.3 CAP

The CAP theorem discusses three aspects:

- Consistency: The data is always the same in every replication on every server.
- Availability: The data must always be accessible (permanently available).
- Partition Tolerance: The database works fine despite network and machine failures.

The theorem says that only two of these aspects can be guaranteed at the same time in a distributed system. You basically have to "pick" two of them. In this paper, we will not discuss the proof of this theorem, we will just accept it as a matter of fact.

3 Solving the problems of SQL with NoSQL

In some applications, NoSQL seems to be more appropriate as a relational SQL database because of some serious problems with SQL. Most of the problems of SQL databases are related to their performance in distributed systems. We will now discuss different ways of solving the performance problems of SQL with NoSQL.

3.1 Scaling

We are all using the Web 2.0. With its huge amount of data going into petabytes (10^{15} bytes), the relational data model is no longer appropriate because it cannot handle it. A possible solution to this performance problem is to scale the database. This can be done in two different ways.

SQL databases mainly use vertical scalability, meaning that a performance advantage is only enabled by increasing the performance of the database server itself. This is also called *up* *scaling*. However, huge amounts of data cannot be handled by just one server that is more powerful because the hardware (e.g. card slots, operating system) is sometime limited.

On the other side the horizontal scaling allows the addition of different distributed database servers for a better performance. This is also called *out scaling*. Note: Distribution alone does not automatically increase the performance of the database! If the database is still managed by one server in the network, then the performance, e.g. number of writes/reads per seconds, does not improve. A horizontal scalability is only given if the computing of queries is also distributed, so that the performance really improves.

3.2 Replication

Horizontal scaling is accompanied by a distribution of data - this leads to data replication. Replications are instances of a resource, e.g. the same file is stored on different servers. The replication can be asynchronous meaning "updated after a transaction" or synchronous meaning "updated during a transaction". For more information about replication in SQL databases please check the seminar papers from 2008 "Database Replication with Oracle 11g and MS SQL Server 2008" [Bolf08] or "Database Replication with MySQL and PostgreSQL" [Mauc08].

4 Categories of NoSQL databases

In the context of this paper we will take a closer look at the four core NoSQL databases. This chapter will also give one example for each category. Some databases belong to more than one category or some sources classify the same database into another category. This paper tries to allocate the most suitable category to each database.

Of course there are more than these four categories, for example object-oriented databases or XML databases. These categories will be briefly discussed further into this paper.

4.1 Key/Value databases

A Key/Value database (K/V) simply has a key and a corresponding value together as dataset. The keys should be unique i.e. occurring just once, what is realized with hashes. Values can be of different types like strings, integers, floats or byte arrays. Some databases do not even care about the data type of the value they just store what you feed. These inputs are so called blobs, meaning an arbitrary binary object, which the database does not need to interpret.

In programming language a key/value pair is also called an associative array, hash-table or hash-map [WiAA10].

Many well-known Web 2.0 applications such as Facebook, use K/V-store. This is simply due to the fact that the datasets are not related to each other and that the data is unstructured. A key/value pair might look as described in figure 1.

Key	Value
f911c336e5b89e	**Content-Type:** image/gif R0lGODlhKwIlAXcAMSH+GlNvZnR3YXJlOiBNaWZ AhsBgQAAAAAAAP8AAPhI+py+0Po5y0AiEM3pl1e

Figure 1: Example of a Key/Value pair

Seen as the datasets do not rely on each other in any form, the databases are freely scalable.

4.1.1 Key/Value database on RAM or on hard disk?

Computers and servers normally have much more hard disk storage than RAM. When the whole data is stored in the RAM, the memory space is limited by the size of the RAM. Moreover, it is normally not possible to increase RAM while the server is up and running. However, RAM is much faster at reading and writing than a hard disk because the data is stored volatile. When there is a power loss, all data stored on the RAM is lost. The main advantage of hard disk storage is that the database size is not limited. If the hard disk space becomes too small, you can increase the memory size just by adding more hard disk space while the server is running. As we have seen before, this is called vertical scaling. For replication it also makes sense to add more nodes (slaves or masters). We know that this is called horizontal scaling.

4.1.2 Advantages

Key/Value databases have a simple data model without relations or structures. They are made for quick and efficient data management in distributed systems.

4.1.3 Examples

There are K/V systems that are running on pure RAM such as *memcached* or purely running on hard disks such as Google's *BigTable*. BigTable is a particular case because it runs on a special file system called *Google File System (GFS)* for managing huge amounts of data. BigTable stores its datasets indexed by a column key, a row key and a timestamp. Of course there are systems that have both RAM and hard disk storage, for example *Redis*.

4.2 Document store

Document store databases use entire documents of different types as datasets. Examples of document types are structured human readable data files such as XML-, JSON- or YAML-files. Document databases can be interpreted as particular cases of a K/V database, but the database has to know what kind of document is saved in it and it has to interpret ("understand") it. This is the biggest difference to Key/Value storage. Therefore, in a document database it is the server itself that can operate with the document and not just the client. Ergo it is also possible to directly edit the data in the document on the server side [DocD10].

Document store databases are schema free. The schema of a database is the structure of the datasets (tuples) i.e. the attributes (e.g. surname, address) and their types (e.g. string, integer, date, etc.). So in a schema free database you do not have such a structure in your database, but rather different structures in the documents. For the storage this does not seem to be a problem, because the documents do not depend on each other. When the documents have some kind of structure or schema it is called semi-structured.

4.2.1 Advantages

The database schema is no longer fixed, so the documents can have arbitrary structures and attributes associated with them. For illustration, let us take one document for a person whose attributes are: name = "John Doe" and webpage = "www.example.net". Let us take another document with the following attributes: name = "Jane Doe", e-mail = "jane@example.net" and children = ("Sarah", "Catherine", "Marcus"). In a SQL database, each one of these attributes must be defined in a dataset (name, webpage, e-mail, child 1, child 2, child 3). For John, e-mail and the children fields are empty (NULL) and for Jane, webpage is empty, so there is an unnecessary overhead. In document store databases, only the attributes that are really used are defined (John: name, webpage; Jane: name, e-mail and children).

In a table of a RDBMS, it is inconceivable that tuples have different attributes.

4.2.2 Example: CouchDB

The name CouchDB is intended to tell you to relax on a couch while installing and using the database, because it is based on well-known principles and even novices would not have any issues installing and using CouchDB. Seen as this database is tolerant to network and other failures, you can use the ironic acronym "Cluster of unreliable commodity hardware database" as well [CIBM09]. This database is open source, written in Erlang and developed within the Apache software foundation.

CouchDB naturally provides a replication management, which only applies changes incrementally, so called incremental replication. The number of distributed independent replica is unlimited, while the applications above have full access to the CRUD-operations (create, read, update, delete) thanks to MVCC (Multi-Version Concurrency Control), meaning no locking of datasets [Intr10].

The ACID concept is fully featured by CouchDB, therefore the database is always consistent after a transaction [Tech10].

CouchDB uses Google's Map/Reduce for querying the database [Edli10]. Thanks to parallel execution, Map/Reduce is a very efficient method to query a huge amount of data on distributed computers. Abstracted it maps a list of values (<key, value> pairs) into another list of values (<key, value> pairs), which is the intermediate result. This list now gets grouped by the key and reduced into one list of values, which is the end result [MaRe10].

CouchDB works with JSON (JavaScript Object Notation) as documents. In figure 2 you can see an example of a JSON document in CouchDB.

```
{
   "_id": "a26ab886a7d200a93a32ae192200416d",
   "_rev": "4-c97d88dfa8c79468bcbe03fcf0e5921d",
   "another_attribute": 17,
   "yet_another_attribute": "a_string",
   "_attachments": {
       "Screenshot.png": {
           "content_type": "image/png",
           "revpos": 3,
           "length": 22666,
           "stub": true
       },
       "trace.nam": {
           "content_type": "application/octet-stream",
           "revpos": 2,
           "length": 7830,
           "stub": true
       }
   }
}
```

Figure 2: Example of a JSON document in CouchDB

Datasets have docIDs and Sequence IDs to distinguish one from another. The Sequence ID changes with every update of the document [Tech10].

The database can be managed through a web interface (RESTful HTTP). There are also interfaces for many scripting languages such as JavaScript, PHP or Perl.

4.3 Column-oriented databases

The basic idea behind column-oriented databases is that not a dataset with its attributes is stored in one unit (row oriented) like in SQL databases but one attribute of a set of datasets is stored in one unit (column oriented). Let us say for an article we have three attributes like the price, the name and the type of an article. Now we have three datasets: 35, Breakfast at Tiffany's, movie; 15, Animal Farm, book; 25, HIStory, music. In column-oriented database we would arrange these datasets as follows: 35, 15, 25; Breakfast at Tiffany's, Animal Farm, HIStory; movie, book, music.

4.3.1 Advantages

So called OLAP (Online Analytical Processing, like Oracle Essbase) or data warehouses are using column-oriented databases. The needed aggregation (computing of a big amount of data to one result, like average value) can be done very quickly in column databases, because all values of the same attribute are successively. In the example above we can aggregate the values to avg(35, 15, 25).

Furthermore like in Key/Value systems there are no relations between datasets.

4.3.2 Example: HBase

HBase is an open source project, written in Java and developed by the Apache software foundation. It tries to model itself after Google's BigTable, therefore it is an open source implementation of the latter. But HBase counts itself among the column-oriented databases and not like BigTable to Key/Value as discussed above. We can see it is elementary for NoSQL databases that they not always belong to one single category, but sometimes to different.

HBase works with Apache's *Hadoop Distributed File System* (HDFS) as basic data storage. It is optimized for high performance real time queries of very huge amounts of distributed data, while datasets are edited seldom but very often inserted. HBase provides a RESTful web service interface supporting XML [HBas10].

4.3.3 Note

The NoSQL archive [Arch10] and [Edli10] mention so called wide column or column family databases, which are basically the same as column-oriented databases. The owner of the NoSQL archive and one of the authors of the book is Stefan Edlich, so basically it is twice the same source. Because there is no "official" source of NoSQL databases this paper partially rests upon the technical literature by Edlich.

Some other sources name column databases "Google BigTable clones" [Kauh10]. Under Wikipedia you will find the same databases under the name column-oriented databases [WiCD10] and this is why the chapter has this name.

4.4 Graph databases

Graph databases originated from the graph theory, where you have vertices (singular: vertex) and edges connecting the vertices. In databases vertices are entities like persons and edges are relations between entities. These relations are similar to the relations in RDBMS [Edli10].

The edges can have properties like a metric of a communication link. Edges can also be used to describe the relation between vertices, like "Fred" - "*knows*" - "Holly". The property "knows" is not directed, so you could also say "Holly" - "knows" - "Fred". Edges of a digraph (directed graph) are then called arrows because the direction is important (e.g. "Holly" - "creation of" - "Truman").

The majority of graph databases are schema free (unstructured) [Edli10]. For storing the context of edges and vertices, directed adjacency lists are used most often. With an adjacency list, the vertex describes to which other vertices it has edges with.

Nowadays, for example, graph databases can be used in location-based services (LBS), to find common friends on social networks or to establish the shortest paths through the daily traffic (with standard algorithms like Dijkstra) or, in a more general manner, for the efficient querying of data in a network.

4.4.1 Advantages

In graph databases, there are relations as in RDBMS, too. If you were to draw or describe them, they would almost be the same. However, in reality there are crucial differences. Relational databases cannot (or only with great effort) change their structure. Queries to various tables or networked data are often too complex in a relational model.

For querying data, graph databases can use so called traversals instead of tedious joins as in SQL databases - this is cheaper and simpler. Graph databases are easy to scale horizontally. The databases use replications of vertices and edges. For an even better performance, the graph theory provides the graph partition. To partition a graph is to divide "a graph into pieces, such that the pieces are of about the same size and there are few connections between the pieces" [WiGP10].

4.4.2 Example: FlockDB

Twitter's FlockDB is interesting because it is easy to understand that a vertex belongs to a person and an edge either has the property "following" (meaning "I am listening to someone's tweets") or "followed" (meaning "Someone listens to my tweets"). So there are two crucial questions for Twitter to ask: "Who follows me?" and "Who do I follow?" To answer these questions, each edge is always stored twice, once with the direction like Sarah - follows - John and once in the other direction John - followed by - Sarah.

FlockDB uses MySQL as the basic database for storing very large directed adjacency lists, "because we understand its behavior - in normal use as well as under extreme load and unusual failure conditions" [TwiE10]. This is another reason for discussing FlockDB. You can see that NoSQL databases sometimes indeed use conventional SQL database systems as their basis, and therefore the term *not only SQL* is justified.

MySQL itself neither supports ACID transactions nor foreign keys [MySA10]. However, there are some database engines such as InnoDB for MySQL which do support transactions and foreign keys [InDB10]. Unfortunately, there is no further information about which MySQL database engine is used for FlockDB. Nevertheless, MySQL is fast at reads and writes.

Figure 3 shows the layout of FlockDB. Flapps (<u>Fl</u>ockDB <u>app</u> servers), written in Scala (<u>sca</u>lable <u>la</u>nguage), are horizontally scalable and stateless, meaning that each request is independent from all other requests. The clients are written in Ruby.

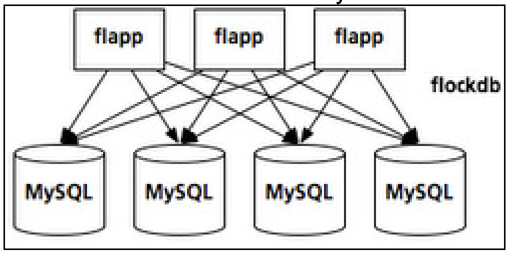

Figure 3: FlockDB Layout [TwiE10]

The partition is done with a Twitter-owned library called Gizzard, but FlockDB is not made for traversals. It is rather built for very quick queries to just one neighbor.

4.5 Other

There are several other databases following a non relational approach, but they are not counted as core NoSQL databases rather as soft NoSQL databases [Edli10].

4.5.1 Object databases

Object databases use the idea of objects in programming languages and map it into database systems.

An example for object databases is db4o for Java and .NET. With db4o you can query the database with program code.

4.5.2 XML databases

XML databases are really non SQL databases. The query language is XQuery, XPath or XUpdate and not SQL anymore. These databases allow storing data in XML format. Therefore the structure of the database is hierarchical. XML is widely used in internet applications so a transformation from (formerly) SQL to XML is not needed anymore.

eXist is a open source Java-based XML database supporting XQuery and XPath and has like CouchDB a RESTful HTTP interface.

4.6 Summary of all examples

Seen as this is not a complete list and because of the multitude of terms and abbreviations used, you can easily lose the overview. Therefore, a summary of the different NoSQL databases is provided at this point (figure 4).

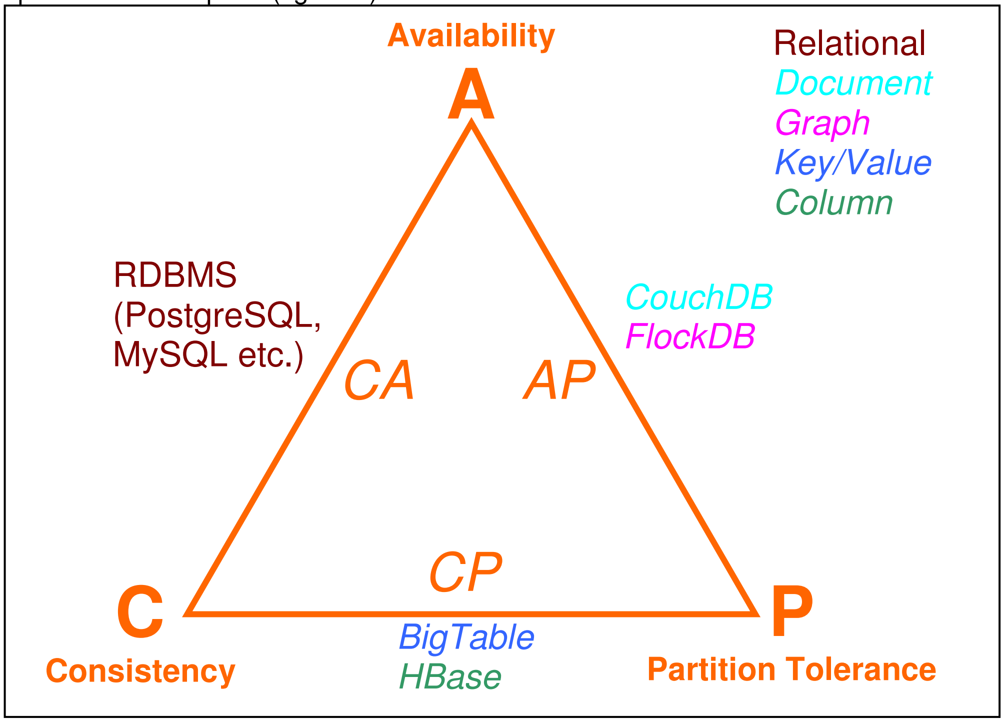

Figure 4: Visualization of CAP theorem with an overview of databases [VisG10]

5 Conclusion

The typical application of NoSQL databases is the Web 2.0 with its huge amount of distributed data and the increasing absence of relations between datasets. Especially in 2009 the number of SQL-avoiding databases exploded what lead into the NoSQL movement. And the number of NoSQL products is still becoming higher.

NoSQL and NoSQL databases as terms are not uniquely defined and many different database systems name themselves NoSQL. There are four main categories (key/value, document, graph and column) and other sub-categories such as object and XML databases.

The main advantages of NoSQL systems are the distribution of the data and the good scalability of the databases.

References

[Arch10]	NoSQL archive, NoSQL Databases, [http://nosql-database.org/], 2010 (retrieved 30 September 2010)
[BigT06]	Google Inc., Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, Robert E. Gruber, [http://static.googleusercontent.com/external_content/untrusted_dlcp/labs.google.com/de//papers/bigtable-osdi06.pdf], 2006 (retrieved 11 November 2010)
[Bolf08]	University of Applied Sciences Chur, Flavio Bolfing, Database Replication with Oracle 11g and MS SQL Server 2008, [http://wiki.hsr.ch/Datenbanken/files/Bolfing_Replication_MSSQL_ORACLE_Paper.pdf], 19 December 2008 (retrieved 13 November 2010)
[CIBM09]	IBM, Joe Lennon, Exploring CouchDB, A document-oriented database for Web applications, [http://www.ibm.com/developerworks/opensource/library/os-couchdb/], 31 March 2009 (retrieved 1 December 2010)
[Disk10]	Wikipedia, Diskussion:NoSQL, Non SQL vs. NoSQL, [http://de.wikipedia.org/w/index.php?title=Diskussion:NoSQL&oldid=80562054#Non_SQL_vs._NoSQL], 21 October 2010 (retrieved 21 October 2010)
[Dist08]	Gregory Chockler (IBM Haifa Research Laboratory), Idit Keidar (Technion), Rachid Guerraoui (EPFL), Marko Vukolic (EPFL), Reliable Distributed Storage, [http://webee.technion.ac.il/~idish/ftp/CGKV-IEEE-Computer08.pdf], 6 October 2008 (retrieved 25 November 2010)
[DocD10]	Ayende Rahien, That No SQL Thing - Document Databases [http://ayende.com/Blog/archive/2010/04/11/that-no-sql-thing-ndash-document-databases.aspx], 11 April 2010 (retrieved 30 November 2010)
[Edli10]	S. Edlich, A. Friedland, J. Hampe, B. Brauer, “NoSQL; Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken”, 2010, Carl Hanser Verlag München, ISBN: 978-3-446-42355-8
[HBas10]	The Apache Software Foundation, Apache HBase, Welcome to HBase!, Overview, [http://hbase.apache.org/#Overview], 6 October 2010 (retrieved 9 December 2010)
[InDB10]	Oracle Corporation, MySQL Documentation, The InnoDB Storage Engine, [http://dev.mysql.com/doc/refman/5.0/en/innodb-storage-engine.html], 2010 (retrieved 13 November 2010)
[Intr10]	The Apache Software Foundation, Apache CouchDB, Introduction [http://couchdb.apache.org/docs/intro.html], 2010 (retrieved 1 December 2010)
[Kauh10]	Harri Kauhanen, NoSQL databases [http://www.slideshare.net/harrikauhanen/nosql-3376398], 9 March 2010 (retrieved 20 November 2010)
[MaRe10]	Saliya Ekanayake, MapReduce in Simple Terms, [http://www.slideshare.net/esaliya/mapreduce-in-simple-terms], April 2010 (retrieved 13 November 2010)
[Mauc08]	University of Applied Sciences Rapperswil, Fabian Mauchle, Database Replication with MySQL and PostgreSQL, [http://wiki.hsr.ch/Datenbanken/files/Mauchle_Replication_MySQL_Postgres_Paper.pdf], 18 December 2008 (retrieved 13 November 2010)
[MySA10]	MySQL is Not ACID Compliant, Alex Popescu, [http://nosql.mypopescu.com/post/1085685966/mysql-is-not-acid-compliant], 8 September 2010 (retrieved 15 November 2010)
[Tech10]	The Apache Software Foundation, Apache CouchDB, Technical Overview [http://couchdb.apache.org/docs/overview.html], 2010 (retrieved 1 December 2010)
[TwiE10]	Twitter engineering, Robey Pointer, Nick Kallen, Ed Ceaser, John Kalucki , Introducing FlockDB, [http://engineering.twitter.com/2010/05/introducing-flockdb.html], 3 May 2010 (retrieved 22 November 2010)
[VisG10]	Nathan Hurst's Blog, Nathan Hurst, Visual Guide to NoSQL Systems, [http://blog.nahurst.com/visual-guide-to-nosql-systems], 15 March 2010 (retrieved 10 November 2010)
[WiAA10]	Wikipedia, Associative array, [http://en.wikipedia.org/wiki/Associative_array], 31 October 2010 (retrieved 11 November 2010)
[WiCD10]	Wikipedia, Column-oriented DBMS, [http://en.wikipedia.org/wiki/Column-oriented_DBMS] 9 December 2010 (retrieved 10 December 2010)
[WiGP10]	Wikipedia, Graph partition, [http://en.wikipedia.org/wiki/Graph_partition], 26 September 2010 (retrieved 9 December 2010)