# NOSQL DATABASES

Answer the questions below by reading indicated sections of the paper "NoSQL Databases" that can be downloaded from:
http://www.christof-strauch.de/nosqldbs.pdf

## Chapter 2

1. What are some of the reasons that led to the development of NoSQL databases? Write your answer in bullet points and explain each point briefly.

Avoidance of Unneeded Complexity Relational databases provide a variety of features and strict data consistency. But this rich feature set and the ACID properties implemented by RDBMSs might be more than necessary for particular applications and use cases

High Throughput Some NoSQL databases provide a significantly higher data throughput than traditional RDBMSs. For instance, the column-store Hypertable which pursues Google's Bigtable approach allows the local search engine Zvent to store one billion data cells per day. To give another example, Google is able to process 20 petabyte a day stored in Bigtable via it's MapReduce approach.

Horizontal Scalability and Running on Commodity Hardware "Definitely, the volume of data is getting so huge that people are looking at other technologies", says Jon Travis, an engineer at SpringSource. Blogger Jonathan Ellis agrees with this notion by mentioning three problem areas of current relational databases that NoSQL databases are trying to address:
1. Scale out data (e. g. 3 TB for the *green badges* feature at Digg, 50 GB for the inbox search at Facebook or 2 PB in total at eBay)
2. Performance of single servers
3. Rigid schema design

## Chapter 3

1. Explain the three parts of CAP briefly. What is the CAP theorem?

**Consistency** meaning if and how a system is in a consistent state after the execution of an operation. A distributed system is typically considered to be consistent if after an update operation of some writer all readers see his updates in some shared data source. (Nevertheless there are several alternatives towards this strict notion of consistency as we will see below.)

**Availability** and especially high availability meaning that a system is designed and implemented in a way that allows it to continue operation (i. e. allowing read and write operations) if e. g. nodes in a cluster crash or some hardware or software parts are down due to upgrades.

**Partition Tolerance** understood as the ability of the system to continue operation in the presence of network partitions. These occur if two or more "islands" of network nodes arise which (temporarily or permanently) cannot connect to each other. Some people also understand partition tolerance as the ability of a system to cope with the dynamic addition and removal of nodes (e. g. for maintainance purposes; removed and again added nodes are considered an own network partition in this notion).

2. For large scale data, for example the data available on the internet, which of the three CAP properties are more important? What is BASE approach and how is it different from ACID approach? Write your answers in bullet points, as far as possible.

Now, Brewer alleges that one can at most choose two of these three characteristics in a "shared-data system". In his talk, he referred to trade-offs between ACID and BASE systems (see next subsection) and proposed as a decision criteria to select one or the other for individual use-cases: if a system or parts of a system have to be consistent and partition-tolerant, ACID properties are required and if availability and partition-tolerance are favored over consistency, the resulting system can be characterized by the BASE properties. The latter is the case for Amazon's Dynamo, which is available and partition-tolerant but not strictly consistent, i. e. writes of one client are not seen immediately after being committed to all readers. Google's Bigtable chooses neither ACID nor BASE but the third CAP-alternative being a consistent and available system and consequently not able to fully operate in the presence of network partitions. In his keynote Brewer points out traits and examples of the three different choices that can be made according to his CAP-theorem.

3. What are some techniques for partitioning data that cannot be stored on a single machine?

Assuming that data in large scale systems exceeds the capacity of a single machine and should also be replicated to ensure reliability and allow scaling measures such as load-balancing, ways of partitioning the data of such a system have to be thought about. Depending on the size of the system and the other factors like dynamism (e. g. how often and dynamically storage nodes may join and leave) there are different approaches to this issue:

**Memory Caches**
**Clustering**
**Separating Reads from Writes**
**Sharding**

# Chapter 5

1. Read about MongoDB and understand and briefly explain the following terms:
- Collections
- Documents

Collections inside databases are referred to by the MongoDB manual as "named groupings of documents". As MongoDB is schema-free the documents within a collection may be heterogeneous although the MongoDB manual suggests to create "one database collection for each of your top level objects". Once the first document is inserted into a database, a collection is created automatically and the inserted document is added to this collection. Such an implicitly created collection gets configured with default parameters by MongoDB—if individual values for options such as auto-indexing, preallocated disk space or size-limits are demanded, collections may also be created explicitly by the createCollection-command.

2. Understand the following query operations using MongoDB. No need for any written answer.
- Selection
- Projection
- Result processing
- Insert
- Update
- Delete
- Aggregation

# Chapter 6

1. Read section 6.1 and write down in points format some of the key features of Bigtable. How has it proved advantageous at Google?

2. Read about the data model of Bigtable and answer the following questions:

- A value in Bigtable is addressed by triplet of row-key, column-key, timestamp.

- Rows are stored in lexicographic order and are dynamically (statically / dynamically) partitioned.

- Number of columns is unlimited (limited / unlimited) and are grouped into units called *column families*.

Timestamps are used to in Bigtable to discriminate different reversion of a cell value

3. Understand HBase, its properties and applications. No written answer required.

4. Write down in points format some key features of Cassandra and its data model.

An instance of Cassandra typically consists of only one table which represents a "distributed multidimensional map indexed by a key". A table is structured by the following dimensions:
**Rows** which are identified by a string-key of arbitrary length. Operations on rows are "atomic per replica no matter how many columns are being read or written".
**Column Families** which can occur in arbitrary number per row. As in Bigtable, column-families have to be defined in advance, i. e. before a cluster of servers comprising a Cassandra instance is launched. The number of column-families per table is not limited; however, it is expected that only a few of them are specified. A column family consists of *columns* and *supercolumns*[12] which can be added dynamically (i. e. at runtime) to column-families and are not restricted in number.
**Columns** have a name and store a number of values per row which are identified by a timestamp (like in Bigtable). Each row in a table can have a different number of columns, so a table cannot be thought of as a rectangle. Client applications may specify the ordering of columns within a column family and supercolumn which can either be by name or by timestamp.
**Supercolumns** have a name and an arbitrary number of columns associated with them. Again, the number of columns per super-column may differ per row.