This document is to share some info collected from web and few features i tried out.

NOSQL

The main principle being 'RDBMS for not all kind of works'. As an article author quotes 'it has to be NOT ONLY SQL rather than NO to SQL'.
So mainly it is giving up some parts of RDDMS for scaling-speed- flexibility-easiness.

Please read this .Informative.
http://www.rackspacecloud.com/blog/2009/11/09/nosql-ecosystem/

What I like about the NoSQL crowd.- Good curtain raiser
http://blog.postmaster.gr/2009/11/05/what-i-like-about-the-nosql-crowd/

A very good discussion at
http://stackoverflow.com/questions/1476295/when-to-use-mongodb-or-other-document-oriented-database-systems

NOSQL storages have 4 formats. 1. Document oriented (for complex data designs) 2. Key value pairs 3. Column -oriented 4. Graph oriented

My understanding is this.

1. Scalability as TB has become common factor even in home servers
2. Neccessity of easy Real time data mining and reports
3. SQL or schema / relational integrity is not required in many designs and its a overhead.
4. RAM has seen drastic downfall in price .4 GB costs around $100
5. We need ripples once in 2 years like RubyonRails, NOSQL, Virtualization, cloud computing
6. Changing data dimensions because of dynamic business model. This is the real kill .Google Buzz is a new BM for Google.
7. Might be a perfect suit for Data warehouse BII systems where 3NF is the design strategy

MongoDB

One of the best web clipping
"Rather than rows in a table, Mongo stores documents in collections.
Documents are (slightly enhanced) JSON objects, so you can stash much more complex structured data in a single document than you can store in a table row.
Natural data structures: arrays, objects, dictionaries. Data modeling becomes a much more natural process."

I . Impressed by this link

http://eu.techcrunch.com//2009/09/08/silentale-lets-you-archive-and-search-your-every-conversation/

II More impressed by this link
http://www.businessinsider.com/how-we-use-mongodb-2009-11

III Caught in glaze of the NOSQL movement (though none of the systems i used in last 5 years scaled more than 3 nodes) , had a goal of get to know about one of the NOSQL DB's.

Though used memcache and java based in memory libraries in previous projects , this is the first time to try a complete NOSQL persistentDB solution.

Deciding between CouchDB and MongoDB to read about was a tougher decision , though both have good list of production deployments by big firms. Selected MongoDB to read about as couchDB is in alpha

http://stackoverflow.com/questions/895762/mongodb-or-couchdb-fit-for-production

http://discuss.joelonsoftware.com/default.asp?biz.5.727383.8

http://www.linkedin.com/answers/technology/information-technology/databases/TCH_ITS_DBS/633444-24053574

Where Can be used:

There is a clear explanation of use cases given in the Mongodb official website
http://www.mongodb.org/display/DOCS/Use+Cases

For more Business cases view the production deployments
http://www.mongodb.org/display/DOCS/Production+Deployments

esp. this is interesting http://www.businessinsider.com/how-we-use-mongodb-2009-11
[One of the websites cofounder seems to a VC for MongoDB]

Additional Info about MongoDB

1. Document store model not table-rows-columns. So nested values is possible with Each keys. Querying will be effective and precise. Uses JSON like Schema.

2. Schemaless- Referential integrity- Joins less. So refactoring and modification is quite easy. For example adding a column to a existing table
http://blog.mongodb.org/post/119945109/why-schemaless

3. Currently Win 32/64 ,Linux 32/64, OS-x 32/64 bit, Unix(*NIX only) versions are available

4. Clusterable. Supports Master - Slave, Replica Pairs model.

5. Supports Shard. (Horizontal Partioning) .Till in Alpha .More details at
http://www.mongodb.org/display/DOCS/Sharding+Introduction

6.Claims better performance than Mysql.
MongoDB performed more than 2X better than MySQL

http://obvioushints.blogspot.com/2009/07/benchmarking-mongodb-vs-mysql.html

7.Has sets of good features like '

- 'updates-in-place' using the Mongo mondifiers- upsert , $inc .
- Capped Collections - Fixed size collections having age out based on insert order.Will be handy in logging & Sorting
- Enhanced Queries

8.Uses JSON for its query and display - BSON binary based
JSON for storing documents for efficiency.

9. ACID in supported for only a single document and cannot span multiples. So not for a banking/Financial application

10. For 64 bit machines the data volume is not limited and on 32 Bit system the map file size can be up to a max of 2 GB

11. Online shell is available at http://try.mongodb.org/

12. Commercial service providers are currently very limited.
The photon PHP based framework is interesting
http://www.mongodb.org/display/DOCS/MongoDB+Commercial+Services+Providers

13. Drivers are available for java.php ,C#,Ruby,C++,Perl

14 Has import - from json string formatted document & export options - in csv

15. MongoDB does not support transactional locking and the primary reasons said to be to avoid deadlocks , distributed locks and to enhance real time performance t has other means of support using techniques - "Update if Current' & 'Insert if Not Present'

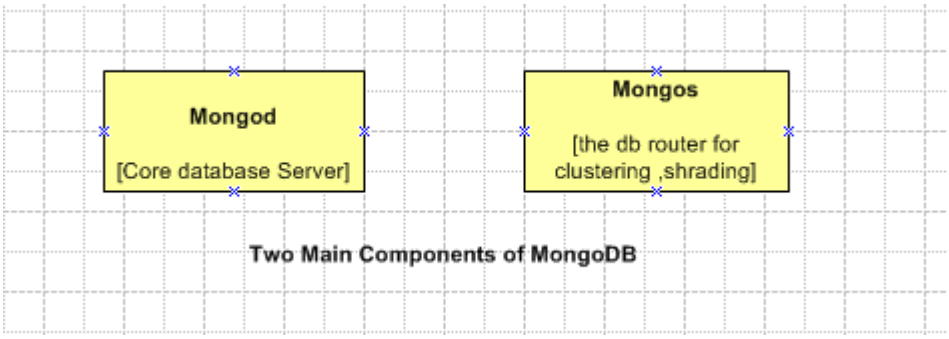
Simple 3 tries using MongoDB (Windows) - java

1. General CRUD using java
2. Image persistence and retrieval
3. Large objects saving and retrieving using GridFS

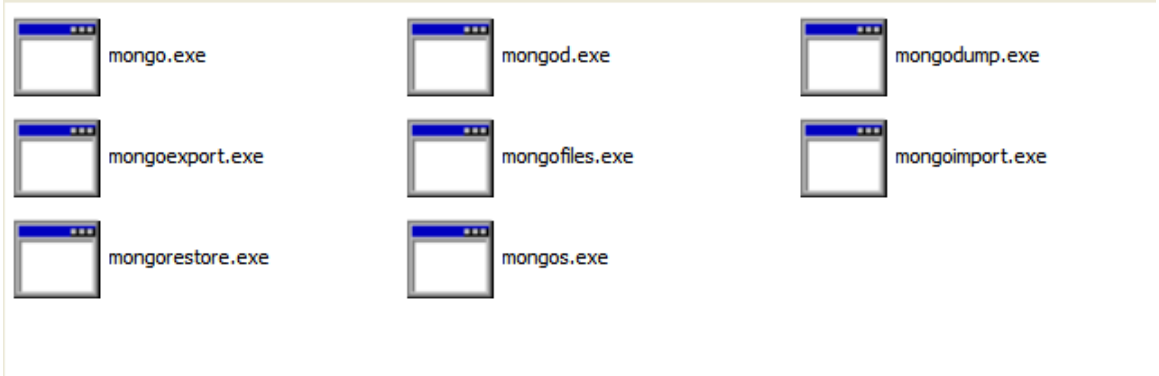
Details about java -MongoDB is available here.
http://www.mongodb.org/display/DOCS/Java+Language+Center

Windows installation is just unzipping the file

[**!** Note create a directory \data\db in the same drive where Mongodb resides]



Files in the bin directory



Server Console

```
D:\mongodb-win32-i386-1.2.4\bin>mongod.exe
mongod.exe --help for help and startup options
Tue Mar 09 14:07:09 Mongo DB : starting : pid = 0 port = 27017 dbpath = /data/db/ master = 0 slave = 0  32-bit

** NOTE: when using MongoDB 32 bit, you are limited to about 2 gigabytes of data
**       see http://blog.mongodb.org/post/137788967/32-bit-limitations for more

Tue Mar 09 14:07:11 db version v1.2.4, pdfile version 4.5
Tue Mar 09 14:07:11 git version: 5cf582d3d96b882c400c33e7670b811ccd47f477
Tue Mar 09 14:07:11 sys info: windows (5, 1, 2600, 2, 'Service Pack 3') BOOST_LIB_VERSION=1_35
Tue Mar 09 14:07:11 waiting for connections on port 27017
Tue Mar 09 14:46:31 connection accepted from 169.254.25.142:1198 #1
Tue Mar 09 14:46:32 building new index on { _id: ObjId(000000000000000000000000) } for mapsDB.Maps...
Tue Mar 09 14:46:32 Buildindex mapsDB.Maps idxNo:0 { name: "_id_", ns: "mapsDB.Maps", key: { _id: ObjId(000000000000000000000000) }
Tue Mar 09 14:46:32 done for 0 records 0.103secs
Tue Mar 09 14:46:32 insert mapsDB.Maps 465ms
Tue Mar 09 14:46:33 MessagingPort recv() error "No error" (0) 169.254.25.142:1198
Tue Mar 09 14:46:33 end connection 169.254.25.142:1198
```

General CRUD using java

Insertion

```
Mongo m = new Mongo( "localhost" , 27017 );
DB db = m.getDB( "callsdb" );
DBCollection coll = db.getCollection("Calldetails");
BasicDBObject doc = new BasicDBObject();
doc.put("call-id", 1);
doc.put("callingparty", "11111111");
doc.put("calledparty", "oneoneone");
coll.insert(doc);
BasicDBObject doc2 = new BasicDBObject();
doc2.put("call-id", 2);
doc2.put("callingparty", "2222222");
doc2.put("calledparty", "TWOTWOTWO");
coll.insert(doc2);
BasicDBObject doc3 = new BasicDBObject();
doc3.put("call-id", 3);
doc3.put("callingparty", "3333333");
doc3.put("calledparty", "THREETHREETHREE");
coll.insert(doc3);
BasicDBObject  doc4 = new BasicDBObject();
doc4.put("call-id", 4);
doc4.put("callingparty", "4444444");
doc4.put("calledparty", "FOURFOURFOUR");
coll.insert(doc4);
```

Insert Result

```
{ "_id" : "4b95f2381038d977297c2f1f" , "call-id" : 1 , "callingparty" : "11111111" , "calledparty" : "oneoneone"}
{ "_id" : "4b95f2381038d9772a7c2f1f" , "call-id" : 2 , "callingparty" : "2222222" , "calledparty" : "TWOTWOTWO"}
{ "_id" : "4b95f2381038d9772b7c2f1f" , "call-id" : 3 , "callingparty" : "3333333" , "calledparty" : "THREETHREETHREE"}
{ "_id" : "4b95f2381038d9772c7c2f1f" , "call-id" : 4 , "callingparty" : "4444444" , "calledparty" : "FOURFOURFOUR"}
```


Searching

```
BasicDBObject query = new BasicDBObject();

 query.put("call-id", 4);

 DBCursor cur = coll.find(query);

 DBObject db_obj=null;
 while(cur.hasNext()) {
    db_obj=  cur.next();
//    System.out.println(db_obj);
     System.out.println(" callingparty "+db_obj.get("callingparty"));
     System.out.println(" calledparty "+db_obj.get("calledparty"));

 }
```

Search Results

```
callingparty 4444444
calledparty FOURFOURFOUR
```

Update

```
BasicDBObject query = new BasicDBObject();
// filtering by call-id 4
query.put("call-id", 4);
DBCursor cur = coll.find(query);

DBObject old_data = null;
// New Record
BasicDBObject new_data = new BasicDBObject();
new_data.put("call-id", 4);
new_data.put("callingparty","***altered--5555");
new_data.put("calledparty", "FIVEFIVEFIVE");

while (cur.hasNext()) {
   old_data = cur.next();
      if (old_data.get("callingparty").equals("4444444")) {
      //updating old with New
         coll.update(old_data, new_data);
   }
```

Result after updating

```
{ "_id" : "4b95f2381038d977297c2f1f" , "call-id" : 1 , "callingparty" : "11111111" , "calledparty" : "oneoneone"}
{ "_id" : "4b95f2381038d9772a7c2f1f" , "call-id" : 2 , "callingparty" : "2222222" , "calledparty" : "TWOTWOTWO"}
{ "_id" : "4b95f2381038d9772b7c2f1f" , "call-id" : 3 , "callingparty" : "3333333" , "calledparty" : "THREETHREETHREE"}
{ "_id" : "4b95f2381038d9772c7c2f1f" , "call-id" : 4 , "callingparty" : "***altered--5555" , "calledparty" : "FIVEFIVEFIVE"}
```

Delete

```
/*To remove call-id 3*/
    BasicDBObject query = new BasicDBObject();
    // filtering by call-id 3
    query.put("call-id", 3);
    DBCursor cur = coll.find(query);

    DBObject old_data = null;


    while (cur.hasNext()) {
       old_data = cur.next();
       // all validation goes here
       coll.remove(old_data);
    }

 }
```

Result

{ "_id" : "4b95f2381038d977297c2f1f" , "call-id" : 1 , "callingparty" : "11111111" , "calledparty" : "oneoneone"}
{ "_id" : "4b95f2381038d9772a7c2f1f" , "call-id" : 2 , "callingparty" : "2222222" , "calledparty" : "TWOTWOTWO"}
{ "_id" : "4b95f2381038d9772c7c2f1f" , "call-id" : 4 , "callingparty" : "***altered--5555" , "calledparty" : "FIVEFIVEFIVE"}

Image persistence and retrieval

```
File file = new File("d:/YellowRose.jpg");
BufferedInputStream buff_ipstream = new BufferedInputStream(
      new FileInputStream(file));
byte[] file_cn = new byte[(int) file.length()];
buff_ipstream.read(file_cn);
buff_ipstream.close();

BasicDBObject record = new BasicDBObject();
record.put("map_id", 1);
// setting image as byte array
record.put("map_image", file_cn);
coll.insert(record);
file_cn = null;
```

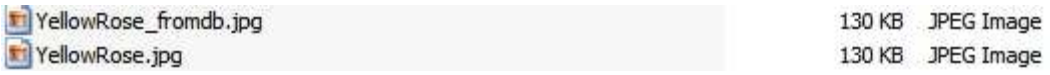
Image Retrieval

```
BasicDBObject search_record = new BasicDBObject();
search_record.put("map_id", 1);
DBCursor cur = coll.find(search_record);
DBObject old_data = null;
byte[] img_fromdb = null;
while (cur.hasNext()) {
   old_data = cur.next();
   img_fromdb = (byte[]) old_data.get("map_image");
}

BufferedOutputStream opStream = new BufferedOutputStream(
      new FileOutputStream("d:/YellowRose_fromdb.jpg"));
opStream.write(img_fromdb);
opStream.flush();
opStream.close();
img_fromdb = null;

}
```

Result



Native storage of BSON supports only 4 MB it seems and 32 bit it is 2 MB i believe as I got an exception below.
Related documentation and info available at
http://www.mongodb.org/display/DOCS/GridFS

```
Exception in thread "main" java.lang.IllegalArgumentException: tried to save too large of an object.   max size : 2098176
        at com.mongodb.ByteEncoder.getTooLargeException(ByteEncoder.java:153)
        at com.mongodb.DBApiLayer$MyCollection.insert(DBApiLayer.java:195)
        at com.mongodb.DBApiLayer$MyCollection.insert(DBApiLayer.java:147)
        at com.mongodb.DBApiLayer$MyCollection.insert(DBApiLayer.java:142)
```

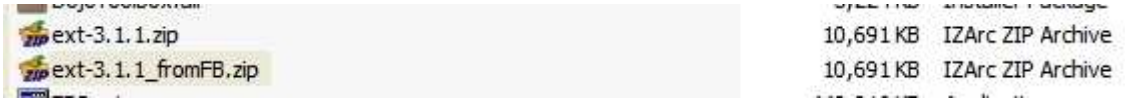
GridFS Insertion

```
// TODO Auto-generated method stub
Mongo m = new Mongo("localhost", 27017);
DB db = m.getDB("filesDB");
File file = new File("d:/ext-3.1.1.zip");

BufferedInputStream buff_ipstream = new BufferedInputStream(
      new FileInputStream(file));
   GridFS gridfs = new GridFS(db);
GridFSInputFile ipfile = gridfs.createFile(buff_ipstream,
      "ext_data_file");
ipfile.save();
buff_ipstream.close();

GridFSDBFile dbfile = gridfs.findOne("ext_data_file");
File file_new = new File("d:/ext-3.1.1_fromFB.zip");
dbfile.writeTo(file_new);
```

GridFS – Retrieval



Links that may provide more information
http://www.linux-mag.com/cache/7530/1.html

http://www.phpclasses.org/blog/post/118-Developing-scalable-PHP-applications-using-MongoDB.html

http://howsoftwareisbuilt.com/2010/02/13/interview-with-eliot-horowitz-cto-of-10gen-mongodb/

http://www.paolocorti.net/2009/12/06/using-mongodb-to-store-geographic-data/