



#### **NoSQL**

**NoSQL Data Models** 

**Cloud Databases** 

Key-value



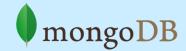


Graph database





Document-oriented





Column family







#### **Motivations**

- Problems with SQL
  - Rigid schema
  - Not easily scalable (designed for 90's technology or worse)
  - Requires unintuitive joins
- Perks of mongoDB
  - Easy interface with common languages (Java, Javascript, PHP, etc.)
  - ■DB tech should run anywhere (VM's, cloud, etc.)
  - Keeps essential features of RDBMS's while learning from key-value noSQL systems



#### Introduction

- A Schema-less / Document Oriented Database
  - Data is stored in documents, not tables / relations
- MongoDB is Implemented in C++ for best performance
- Platforms: 32/64 bit Windows Linux, Mac OS-X, FreeBSD, Solaris
- Language drivers: Ruby/Ruby-on-Rails, Java,
   C#, JavaScript, C / C++, Erlang, Python, Perl...
- Replication & High Availability
- Map/Reduce
- Querying & Fast In-Place Updates



### Why use MongoDB

- Simple queries
- Makes sense with most web applications
- Easier and faster integration of data
- Not well suited for heavy and complex transactions systems.
- Performance / Scalability / Availability
  - No Joins + No multi-row transactions
    - Fast Reads / Writes
  - Async writes
    - · you don't wait for inserts to complete
  - Secondary Indexes
    - Index on embedded document fields for superfast adhoc queries

### Document store: analogy wrt RDBM

RDBMS		MongoDB
Database	$\Rightarrow$	Database
Table, View	$\Rightarrow \Rightarrow$	Collection
Row	$\Rightarrow$	Document (JSON, BSON)
Column		Field
Index	$\Rightarrow$	Index
Join		Embedded Document
Foreign Key		Reference
Partition		Shard

## MongoDB's Data Model

- A MongoDB instance may have zero or more databases
- A Database has "Collections"
  - Collections have "Documents"
    - Documents have "Fields"
      - -Fields are key = value pairs
  - –A Collection does not enforce the structure of its documents\*
  - \*i.e. Schemaless



#### **JSON**

- "JavaScript Object Notation"
- Easy for humans to write/read, easy for computers to parse/generate
- Objects can be nested
- **■** Built on
  - name/value pairs
  - Ordered list of values



#### **BSON**

- "Binary JSON"
- Binary-encoded serialization of JSON-like docs
- Also allows "referencing"
- Embedded structure reduces need for joins
- Goals
  - Lightweight
  - Traversable
  - Efficient (decoding and encoding)



### **BSON Example**

```
"_id":
        "37010"
"city": "ADAMS",
"pop": 2660,
"state": "TN",
"councilman": {
             name: "John Smith"
             address: "13 Scenic Way"
```

# **BSON Types**

Туре	Number
Double	1
String	2
Object	3
Array	4
Binary data	5
Object id	7
Boolean	8
Date	9
Null	10
Regular Expression	11
JavaScript	13
Symbol	14
JavaScript (with scope)	15
32-bit integer	16
Timestamp	17
64-bit integer	18
Min key	255
Max key	127

The number can be used with the \$type operator to query by type!



#### The \_id Field

- By default, each document contains an \_id field.
   This field has a number of special characteristics:
  - Value serves as primary key for collection.
  - Value is unique, immutable, and may be any non-array type.
  - Default data type is ObjectId, which is "small, likely unique, fast to generate, and ordered." Sorting on an ObjectId value is roughly equivalent to sorting on creation time.



# mongoDB vs. SQL

mongoDB	SQL	
Document	Tuple	
Collection	Table/View	
PK: _id Field	PK: Any Attribute(s)	
Uniformity not Required	Uniform Relation Schema	
Index	Index	
Embedded Structure	Joins	
Shard	Partition	



### Installation and Running MongoDB

- 1. Download from mongodb.org
- 2. Unzip
- 3. Create data directory
- >mkdir c:\data\db
- 4. Run MongoDB (mongod):
- >cd c:\mongodb-1.6.3\bin
- >mongod
- 5. Run Mongo shell (mongo):
- >mongo



#### The Mongo Shell

```
>mongo
>help()
>show dbs
>use <dbname>
>show collections
>db.collectionName.findOne()
```

```
>db.collectionName.find()
```

>db.help()

>db.collectionName.help()

```
C:\appservers\mongo-1.6.3\bin\mongo.exe
MongoDB shell version: 1.6.3
connecting to: test
 show dbs
admin
cfmongodb_tests
defau1t_db
local
mongorocks
test
 use mongorocks
switched to db mongorocks
 show collections
people
system.indexes
  db.people.findOne()
         "_id" : ObjectId("4cb66dae636ac4fa2045ff31"),
         "COUNTER" : NumberLong(1),
         "LOVESMONGO" : true,
         "NAME" : "Marc",
         "BIKE" : "Felt",
         "LOVESSQL" : true,
         "KIDS" : [
                             "NAME" : "Alexis",
                             "AGE" : NumberLong(7),
                             "DESCRIPTION" : "crazy",
"HAIR" : "blonde"
                            "NAME" : "Sidney",
"AGE" : NumberLong(2),
"DESCRIPTION" : "ornery",
"HAIR" : "dirty blonde"
          ,
WIFE" : "Heather",
         "TS" : "Wed Oct 13<sup>2010</sup> 22:40:46 GMT-0400 (Eastern Daylight Time)"
```

#### **CRUD**

- Create
  - db.collection.insert( <document> )
  - db.collection.save( <document> )
  - db.collection.update( <query>, <update>, { upsert: true } )
- Read
  - db.collection.find( <query>, , ction> )
- Update
- Delete
  - db.collection.remove( <query>, <justOne> )

# **CRUD** example

```
> db.user.insert({
    first: "John",
    last : "Doe",
    age: 39
})
```

```
> db.user.find ()
{
    "_id" : ObjectId("51..."),
    "first" : "John",
    "last" : "Doe",
    "age" : 39
}
```

```
> db.user.remove({
     "first": /^J/
})
```