open-source, high-performance, document-oriented database

Michael Dirolf
Software Engineer, 10gen
mike@10gen.com
20 April 2010
Non-relational Operational Stores
("NoSQL")

New Gen. OLAP
(vertica, aster, greenplum)

RDBMS
(Oracle, MySQL)
NoSQL Really Means:

non-relational, next-generation operational datastores and databases
Horizontally Scalable Architectures

- no joins
- no complex transactions
New Data Models

- no joins
- no complex transactions
New Data Models

improved ways to develop applications?
Data Models

Key / Value
memcached, Dynamo

Tabular
BigTable

Document Oriented
MongoDB, CouchDB, JSON stores
Focus on performance
  depth of functionality
  scalability & performance

- memcached
- key/value

- RDBMS

mongoDB
JSON-style Documents represented as BSON

```json
{ "hello": "world" }
```

`\x16\x00\x00\x00\x02hello
\x00\x06\x00\x00\x00\x00world
\x00\x00`
Flexible "Schemas"

{“author”: “mike”,
 “text”: “...”}

{“author”: “eliot”,
 “text”: “...”,
 “tags”: [“mongodb”]}
Dynamic Queries
Atomic Update
Modifiers
Focus on Performance
Replication
Auto-sharding
Many Supported Platforms / Languages
Best Use Cases

- Scaling Out
- Caching
- The Web
- High Volume
Less Good At

highly transactional

ad-hoc business intelligence

problems that require SQL
A Quick Aside

_id

special key

present in all documents

unique across a Collection

any type you want
Post

{author: "mike",
date: new Date(),
text: "my blog post...",
tags: ["mongodb", "intro"]}
Comment

{author: "eliot",
date: new Date(),
text: "great post!"}
New Post

post = {author: "mike",
    date: new Date(),
    text: "my blog post...",
    tags: ["mongodb", "intro"]}

db.posts.save(post)
Embedding a Comment

c = {author: "eliot",
    date: new Date(),
    text: "great post!"}

db.posts.update({_id: post._id},
    {$push: {comments: c}})
Posts by Author

db.posts.find({author: "mike"})
db.posts.find()
  .sort({date: -1})
  .limit(10)
Posts Since April 1

last_week = new Date(2010, 3, 1)

db.posts.find({date: {$gt: last_week}})
Posts Ending With ‘Tech’

db.posts.find({text: /Tech$/})
Posts With a Tag

db.posts.find({tags: "mongodb"})

...and Fast

(multi-key indexes)

db.posts.ensureIndex({tags: 1})
Indexing / Querying on Embedded Docs (dot notation)

db.posts.ensureIndex({"comments.author": 1})

db.posts.find({"comments.author": "eliot"})
Counting Posts

db.posts.count()

db.posts.find({'author': 'mike'}).count()
Basic Paging

page = 2
page_size = 15

db.posts.find().limit(page_size)
 .skip(page * page_size)
Migration: Adding Titles

(just start adding them)

define post = {author: "mike",
               date: new Date(),
               text: "another blog post...",
               tags: ["mongodb"],
               title: "MongoDB for Fun and Profit"}

define post_id = db.posts.save(post)
Advanced Queries

$gt, $lt, $gte, $lte, $ne, $all, $in, $nin

db.posts.find({$where: "this.author == 'mike' || this.title == 'foo'"})
Other Cool Stuff

aggregation and map/reduce
capped collections
unique indexes
mongo shell
GridFS
geo