Open source, high performance database

Anti-social Databases: NoSQL and MongoDB

Will LaForest
Senior Director of 10gen Federal
will@10gen.com
@WLaForest
Dawn of Databases to Present

IBM’s IMS founded

SQL invented

Oracle founded

Codd publishes relational model paper in 1970

PC’s gain traction

Client Server

Dynamic Web Content

Web applications

SOA

BigTable

3 tier architecture

Cloud Computing

Brewer’s Cap born

WWW born

10gen founded

NoSQL Movement

IBM’s IMS

Codd publishes relational model paper in 1970

PC’s gain traction

Client Server

Dynamic Web Content

Web applications

SOA

BigTable

3 tier architecture

Cloud Computing

Brewer’s Cap born

WWW born

10gen founded

NoSQL Movement

mongoDB released

mongoDB

mongoDB

mongoDB
### Relational Databases

![Diagram of tuples and relations in relational databases](image-url)

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Relation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Attribute**: Represents the characteristics of a tuple.
- **Relation**: A set of tuples, each tuple having a fixed number of attributes.
Relational Databases

Diagram:
- Category
- BlogCategory
- Author
- Blog
- BlogTag
- Content
- Tag

Relationships:
- N:1
- 1:N

Diagram represents a relational database model with relationships between Category, BlogCategory, Author, Blog, BlogTag, Content, and Tag.
• Data stored in a RDBMS is very compact (disk was more expensive)
• SQL and RDBMS made queries flexible with rigid schemas
• Rigid schemas helps optimize joins and storage
• Massive ecosystem of tools, libraries and integrations
• Been around 40 years!
Enter Big Data (Volume, Variety, Velocity)

Sensor Data/SIGINT (volume, velocity)

Crowd Sourcing (volume, variety)

Asset Management (variety, velocity)

Open Source (volume, variety, velocity)
RDBMS Challenges

DATA VARIETY & VOLATILITY
- Extremely difficult to find a single fixed schema
- Don’t know data schema a-priori

VOLUME & NEW ARCHITECTURES
- Systems scaling horizontally, not vertically
- Commodity servers
- Cloud Computing

TRANSACTIONAL MODEL
- N x Inserts or updates
- Distributed transactions
• Non-relational been hanging around (heard of MUMPS?)
• Modern NoSQL theory and offerings started in early 2000s
• NoSQL = Not Only SQL
• A collection of very different products
• Alternatives to relational databases when a bad fit
• Common motivations
  – Horizontally scalable (commodity server/cloud computing)
  – Schema Flexibility
NoSQL categories

• I group by data model
  – Some people use or include data arrangement

• Data arrangement is important and cuts across data models (I have separate talk on this)
  – Column/Document
  – Consistent hashing partitioning
  – Range based partitioning

• Key/Value
• Big Table Descendents
• Document Oriented
• Graph
NoSQL Databases – Key/Value

- Value (Data) mapped to a key (think primary)
- Some typed, some just BLOBs
- Fast hash based queries
- No range queries, no ordering, simple data model

```
"Will"
- "will@10gen.com"

"Chris"
- "chris@10gen.com"

Will-obj
- [4e61 6d65 3a57 ...]
NoSQL Databases – Big Table Descendents

- Data stored on disk in a column oriented fashion
- Predominantly hash based indexing
- Rudimentary or no secondary indexes
- Range queries and ordering on one dimension (row key)
- Some consistent hashing some range based

<table>
<thead>
<tr>
<th>row keys</th>
<th>column family “contact”</th>
<th>column family “personal”</th>
</tr>
</thead>
<tbody>
<tr>
<td>row</td>
<td>Will</td>
<td>“bio”: “Will attended … ” “picture”: …</td>
</tr>
<tr>
<td></td>
<td>“twitter”: “wlaforest”</td>
<td>“email”: “wlaforest”</td>
</tr>
<tr>
<td></td>
<td>“email”: “<a href="mailto:will@10gen.com">will@10gen.com</a>”</td>
<td>“phone”: “555-555-5555”</td>
</tr>
<tr>
<td>row</td>
<td>Chris</td>
<td>“bio”: “… ” “picture”: “…” “hobby”: “golf”</td>
</tr>
<tr>
<td></td>
<td>“email”: “<a href="mailto:chris@10gen.com">chris@10gen.com</a>”</td>
<td>“phone”: “555-555-5555”</td>
</tr>
</tbody>
</table>

Example column family “contact”:
- Will attends …
- Picture of Will

Example column family “personal”:
- Golf as a hobby
NoSQL Databases – Document Oriented

- Data modeled as documents or objects (XML and JSON)
- No fixed schema
- Richest data model
- Consistent hashing and range based partitioning

```json
{name: "will",
  eyes: "blue",
  birthplace: "NY",
  aliases: ["bill", "la ciaccio"],
  gender: "???",
  boss: "ben"}

{name: "jeff",
  eyes: "blue",
  height: 72,
  boss: "ben"}

{name: "brendan",
  aliases: ["el diablo"]}

{name: "matt",
  pizza: "DiGiorno",
  height: 72,
  boss: 555.555.1212}

{name: "will",
  eyes: "blue",
  birthplace: "NY",
  aliases: ["bill", "la ciaccio"],
  gender: "???",
  boss: "ben"}

{name: "ben",
  hat: "yes"}

{name: "ben",
  hat: "yes"}
```
What about Hadoop?

- Not a database!
- Map Reduce on HDFS or other data source
- Great for grinding through data
- When you want to use it
  - Can’t use a index
  - Distributing custom algorithms
  - ETL
- Many NoSQL offerings have native map reduce functionality
MongoDB & 10gen
10gen, the MongoDB Company

• 2007 founded
• 2009 first release of MongoDB
• MongoDB is open source
• 10gen has a Redhat-like business model
  – Subscriptions (subscriber build, support)
  – Training
  – Consulting
• > 80M in funding
  – Sequoia, NEA, In-Q-Tel, Union Square Ventures, Flybridge Capital,
MongoDB is the leading NoSQL solution

#2 on Indeed’s Fastest Growing Jobs

![Job Trends from Indeed.com](image)

- HTML5
- MongoDB
- iOS
- Android
- Mobile app
- Puppet
- Hadoop
- jQuery
- PaaS
- Social Media

Jaspersoft BigData Index

Demand for MongoDB, the document-oriented NoSQL database, saw the biggest spike with over 200% growth in 2011.

Google Searches

![Google Insights for Search](image)

451 Group

“MongoDB increasing its dominance”

![Relative adoption of NoSQL - LinkedIn member skills](image)
MongoDB is the leading NoSQL solution

#2 ON INDEED’S FASTEST GROWING JOBS

<table>
<thead>
<tr>
<th>Top Job Trends</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. HTML5</td>
<td></td>
</tr>
<tr>
<td>2. MongoDB</td>
<td></td>
</tr>
<tr>
<td>3. iOS</td>
<td></td>
</tr>
<tr>
<td>4. Android</td>
<td></td>
</tr>
<tr>
<td>5. Mobile app</td>
<td></td>
</tr>
<tr>
<td>6. Puppet</td>
<td></td>
</tr>
<tr>
<td>7. Hadoop</td>
<td></td>
</tr>
<tr>
<td>8. jQuery</td>
<td></td>
</tr>
<tr>
<td>9. PaaS</td>
<td></td>
</tr>
<tr>
<td>10. Social Media</td>
<td></td>
</tr>
</tbody>
</table>
MongoDB is the leading NoSQL solution.

“MongoDB INCREASING ITS DOMINANCE”
Different Assumptions

• Scale horizontally over commodity hardware
• Agility essential (schema free/heterogeneous interface)
• RDBMSs great so keep what works
  – Rich data models
  – Adhoc queries
  – Fully featured indexes
• What doesn’t distribute well?
  – Long running multi-row transactions
  – Join
  – Both artifacts of the relational data model
MongoDB Key Features

- Data stored as documents (JSON)
- Schema free
- CRUD operations – (Create Read Update Delete)
- Atomic document operations
- Consistent (but is tunable... advanced topic)
- Rich indexing (secondary, geospatial, covered)
- Ad hoc Queries like SQL
  - Equality
  - Ranges
  - Regular expression searches
  - Geospatial
- Replication – HA, read scalability, geo centric reads
- Sharding (sometimes called partitioning) for scalability
<table>
<thead>
<tr>
<th>RDBMS</th>
<th>MongoDB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td>Database</td>
</tr>
<tr>
<td>Table</td>
<td>Collection</td>
</tr>
<tr>
<td>Row</td>
<td>Document</td>
</tr>
</tbody>
</table>
Usage Examples
var p = {
  author: "roger",
  date: new Date(),
  text: "Spirited Away",
  tags: ["Tezuka", "Manga"]
}

> db.posts.save(p)
>db.posts.find()

```javascript
{
  _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
  author : "roger",
  date : "Sat Jul 24 2010 19:47:11 GMT-0700 (PDT)",
  text : "Spirited Away",
  tags : [ "Tezuka", "Manga" ]
}
```

Notes:
- _id is unique, but can be anything you’d like
Secondary Indexes

Create index on any Field in Document

// 1 means ascending, -1 means descending

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

db.posts.find({author: 'roger'})

{ _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
  author : "roger",
  ... }

mongoDB
Query Operations

• Conditional Operators
  – $all, $exists, $mod, $ne, $in, $nin, $nor, $or, $size, $type
  – $lt, $lte, $gt, $gte

// find posts with any tags
> db.posts.find( {tags: {exists: true}} )

// find posts matching a regular expression
> db.posts.find( {author: /^rog*/i} )

// count posts by an author before a certain date
> db.posts.find( {author: 'roger', date:{ $lt: Sat... }} ).count()
Atomic Operations

- $set, $unset, $inc, $push, $pushAll, $pull, $pullAll, $bit

> comment = { author: “fred”,
  date: new Date(),
  text: “Best Movie Ever”}

> db.posts.update( { _id: “...” },
  $push: {comments: comment} );
Nested Documents

{ 
  _id : ObjectId("4c4ba5c0672c685e5e8aabf3"),
  author : "roger",
  date : "Sat Jul 24 2010 19:47:11 GMT-0700 (PDT)",
  text : "Spirited Away",
  tags : [ "Tezuka", "Manga" ],
  comments : [ 
    { 
      author : "Fred",
      date : "Sat Jul 24 2010 20:51:03 GMT-0700 (PDT)",
      text : "Best Movie Ever"
    }
  ]
}
Secondary Indexes

// Index nested documents
> db.posts.ensureIndex( "comments.author":1 )
  db.posts.find({‘comments.author’:’Fred’})

// Index on tags
> db.posts.ensureIndex( tags: 1)
> db.posts.find( { tags: ’Manga’ } )

// geospatial index
> db.posts.ensureIndex( “author.location”: “2d” )
> db.posts.find( “author.location” : { $near : [22,42] } )
Aggregates, Statistics, and Analytics
Computations & Aggregates Over MongoDB

- Do them client side (Python or R)
- Native Map/Reduce in JS in MongoDB
  - Distributes across the cluster with good data locality
- New aggregation framework
  - Declarative (no JS required)
  - Pipeline approach (like Unix `ps -ax | tee processes.txt | more`)
- Hadoop
  - Intersect the indexing of MongoDB with the brute force parallelization of hadoop
  - Hadoop MongoDB connector
- BI/Reporting Tools
  - Pentaho, Jaspersoft, Ikanow, Nucleon
• Prod clusters on order of
  • 100B objects
  • 50k qps per server
  • ~1400 nodes