

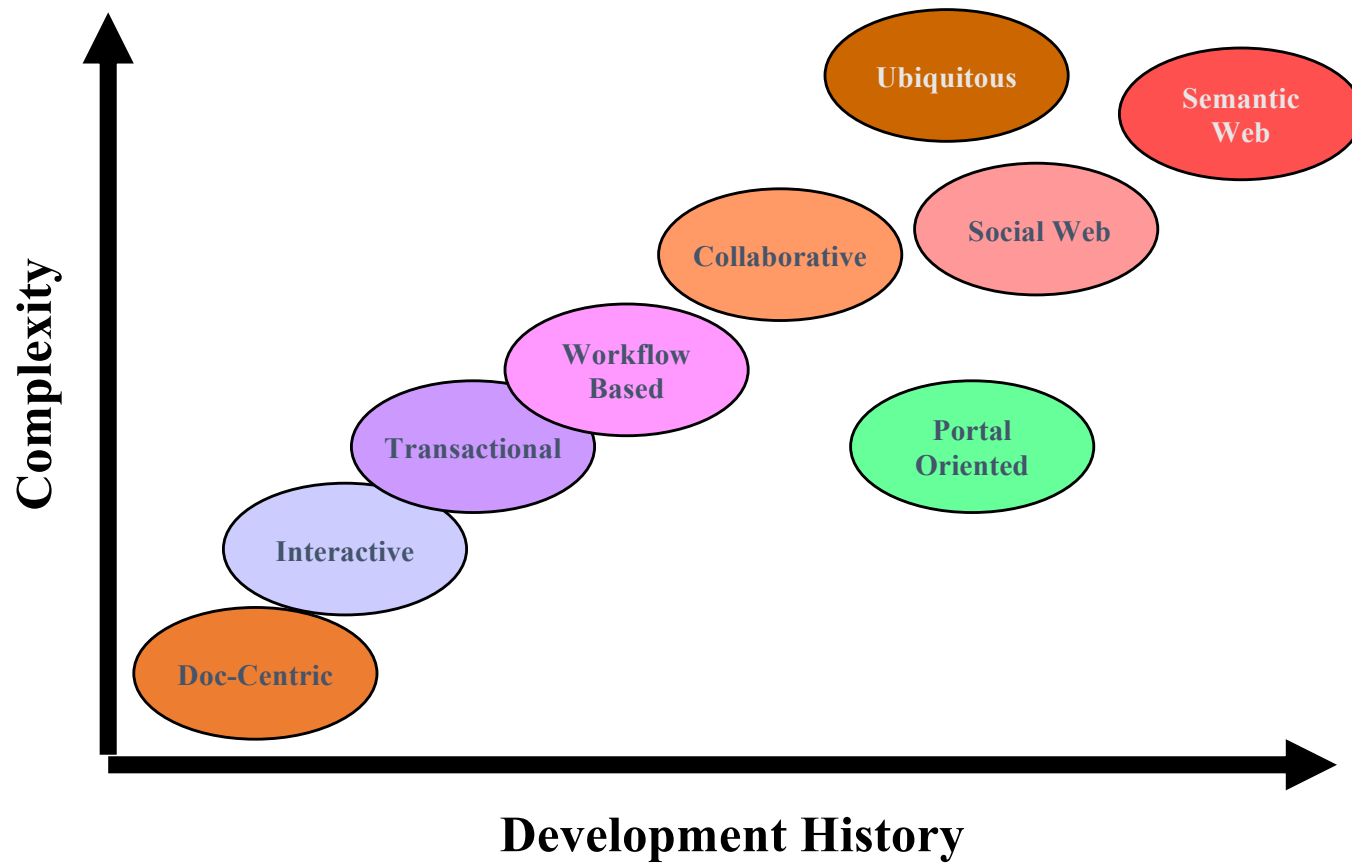
Web Database

- Introduction to Web Databases
- Types of Web Databases
- Database Models in Web Development
- MongoDB

What is web database?

- A web database is a system for storing and displaying information that is accessible from the internet/web.
- A web database is a system for storing information that can be accessed via a website

- Categories of web applications



Architecture Defined

- Define “software architecture”
- <http://www.sei.cmu.edu/architecture/definitions.html>
- “Software architecture is the set of design decisions which, if made incorrectly, may cause your project to be cancelled.” – Eoin Woods
- Authors focus on 5 key attributes of software architectures
 - Structure, Elements, Relationships
 - Analysis => Implementation
 - Multiple viewpoints (conceptual, runtime, process & implementation)
 - Understandable
 - Framework for flexibility

Developing Architectures

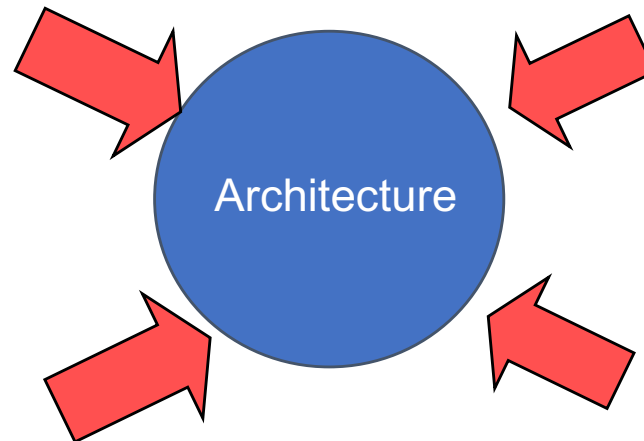
● Influences on Architectures

Functional Requirements

- Clients
- Users
- Other Stakeholders

Experience with

- Existing Architecture
- Patterns
- Project Management
- Other?



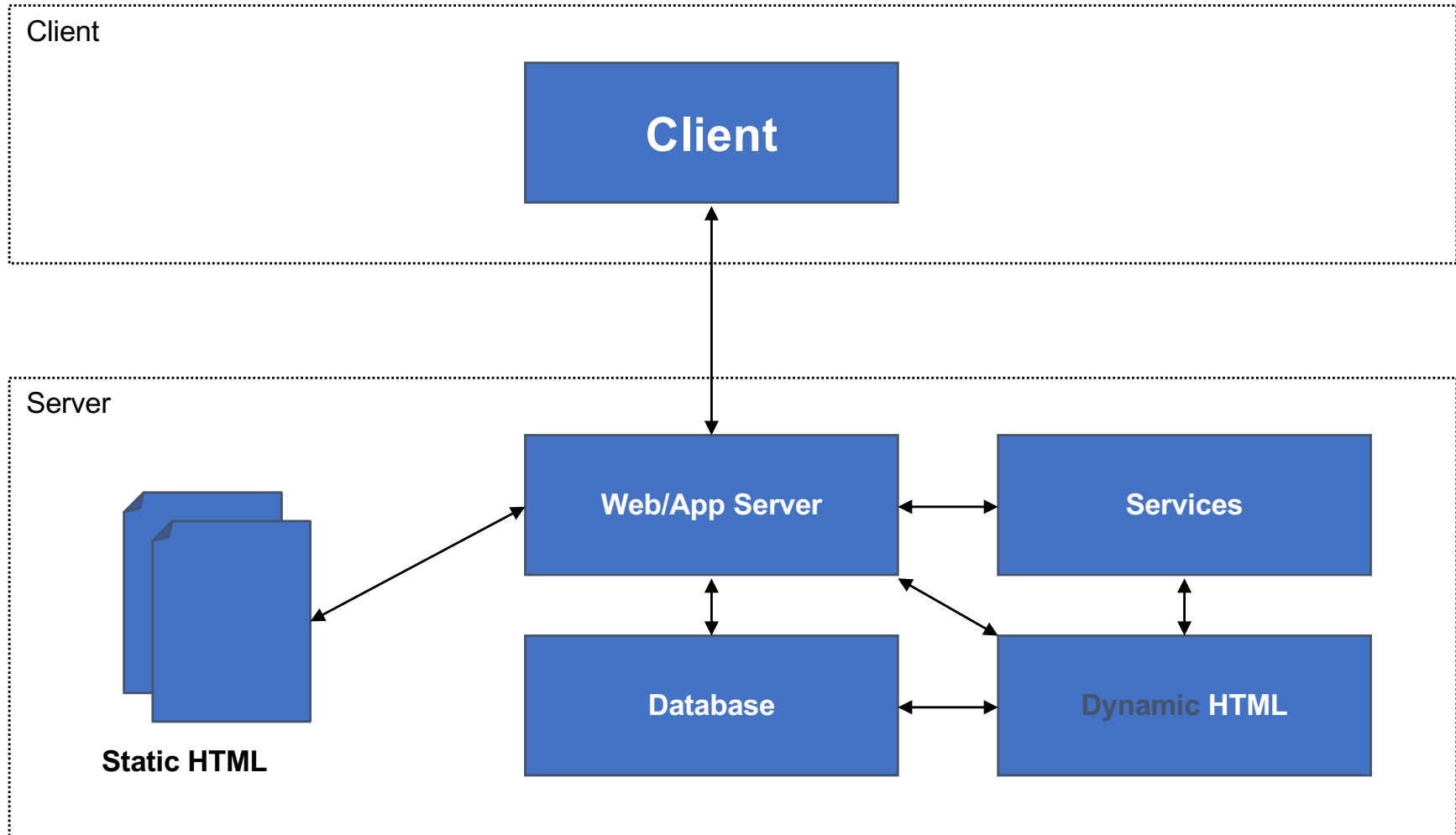
Quality considerations with

- Performance
- Scalability
- Reusability
- Other?

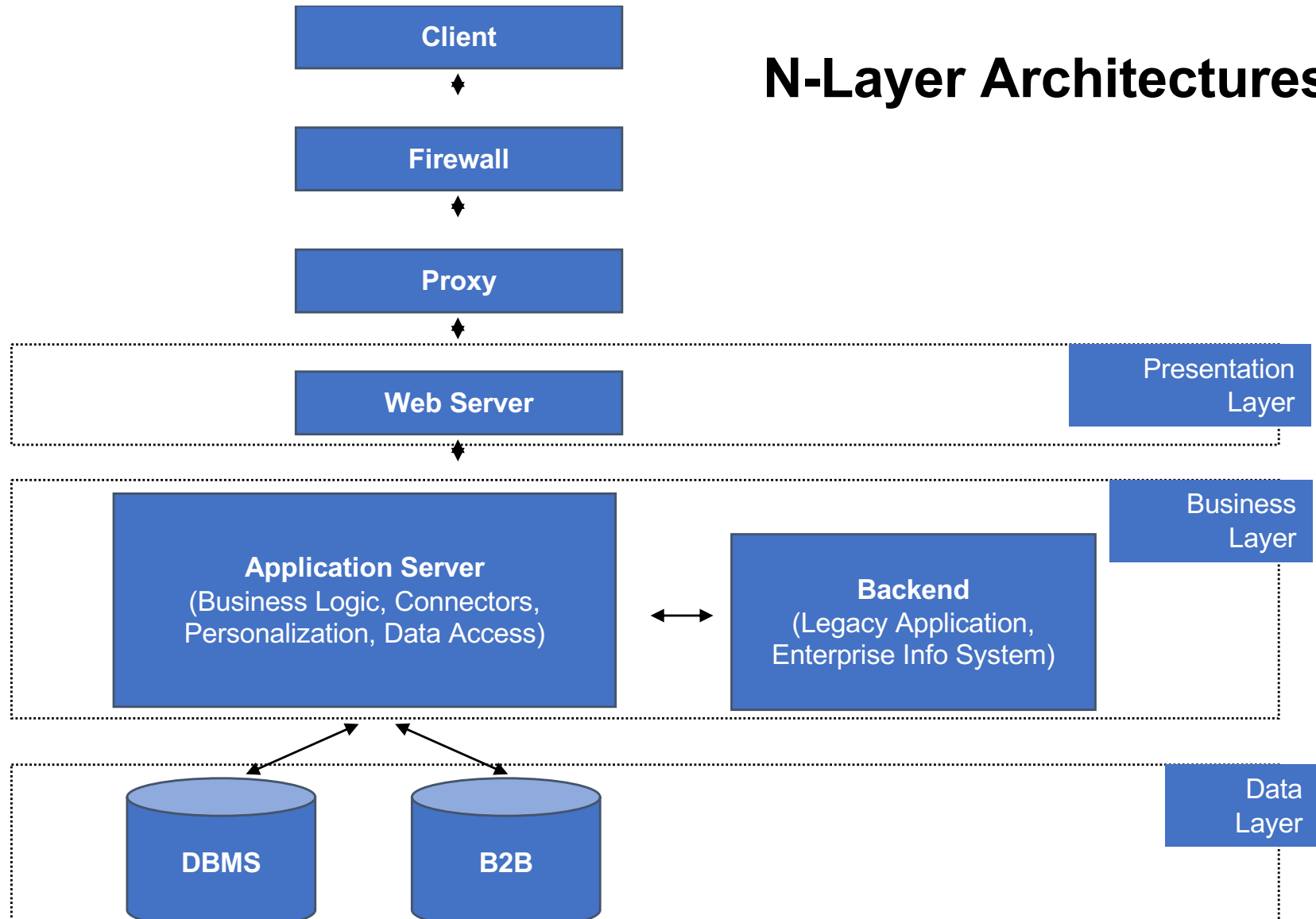
Technical Aspects

- Operating System
- Middleware
- Legacy Systems
- Other?

Client/Server (2-Layer)



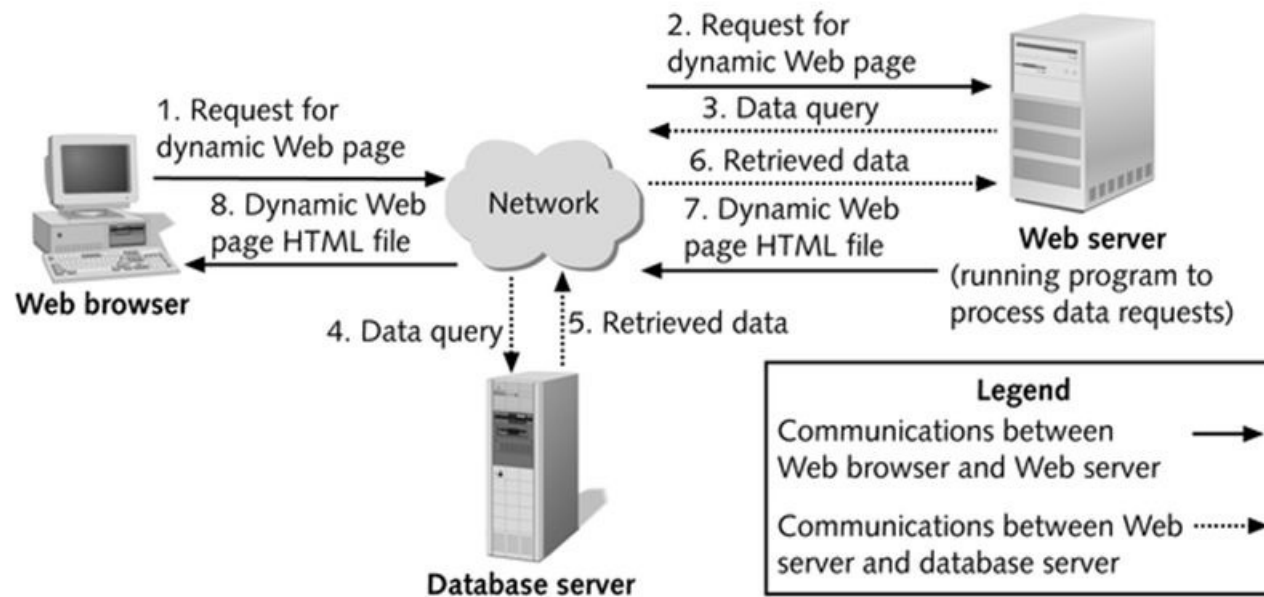
N-Layer Architectures



Why an N-Layer Architecture?

- Separating services in business layer promotes re-use different applications
 - Loose-coupling – changes reduce impact on overall system.
 - More maintainable (in terms of code)
 - More extensible (modular)
- Trade-offs
 - Needless complexity
 - More points of failure

Database-driven website arch.



Types of web databases

- MySQL - **open-source DBMS for relational databases.** – coupled with PHP (**LAMP** stack)
- PostgreSQL
- MongoDB - **stores data as collections of documents**

Database Popularity

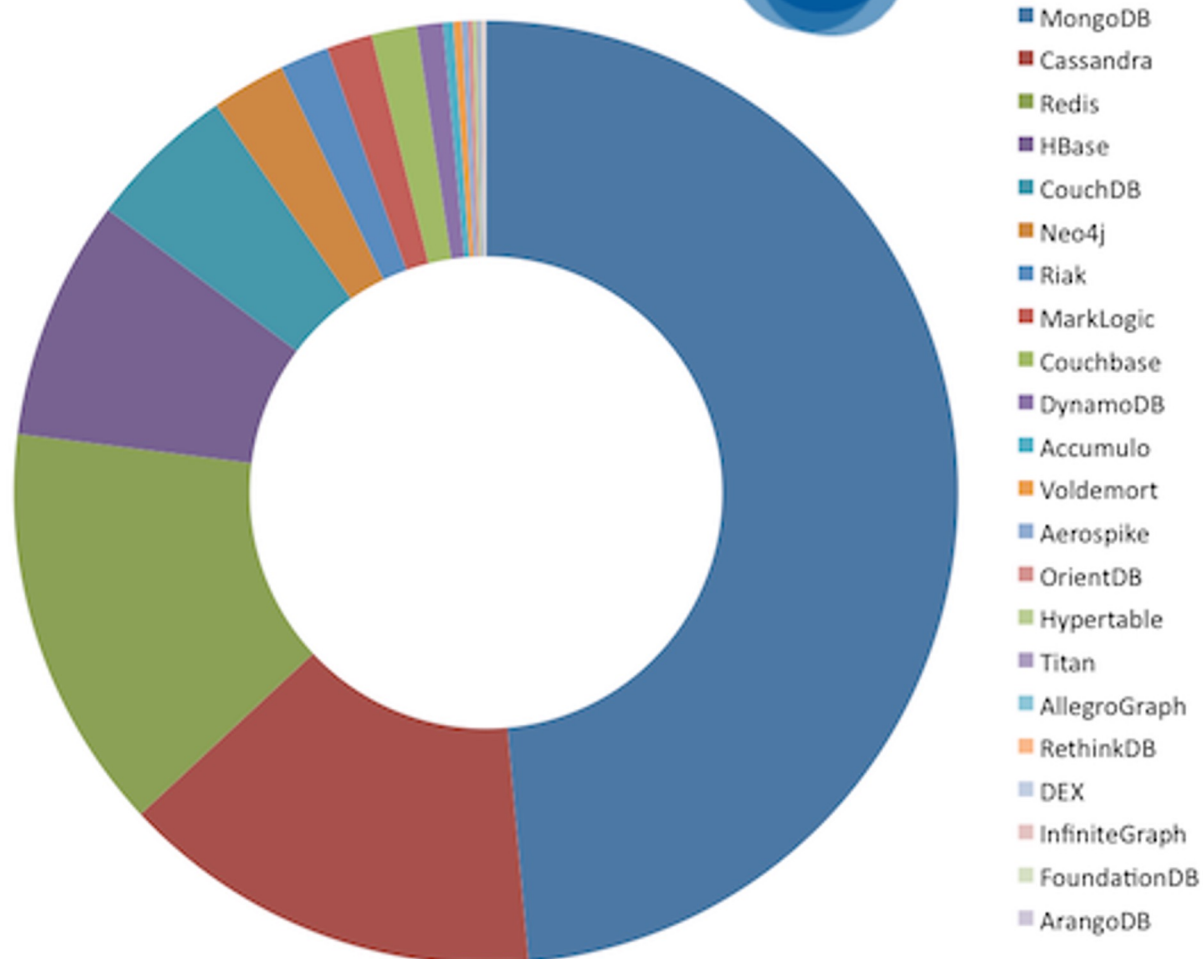
Rank	Name	Score
1.	Oracle	1617.19
2.	MySql	1254.27
3.	SQL Server	1234.46
4.	PostgreSQL	190.83
5.	DB2	165.9
6.	MongoDB	161.87
7.	Microsoft Access	141.6
8.	SQLite	78.78
9.	Sybase	77.75

<http://db-engines.com/en/ranking>

Relative adoption of NoSQL skills -
LinkedIn member search Sept 2013

451

Research



History

- 2004 - Google BigTable Paper
 - NoSql becomes mainstream technology
 - Kicks off movement to create “web scale” databases
- 2007 - 10Gen releases MongoDB
 - Bridges gap between key-values and RDBMS
- 2009 - Open Sourced - [GitHub](#)

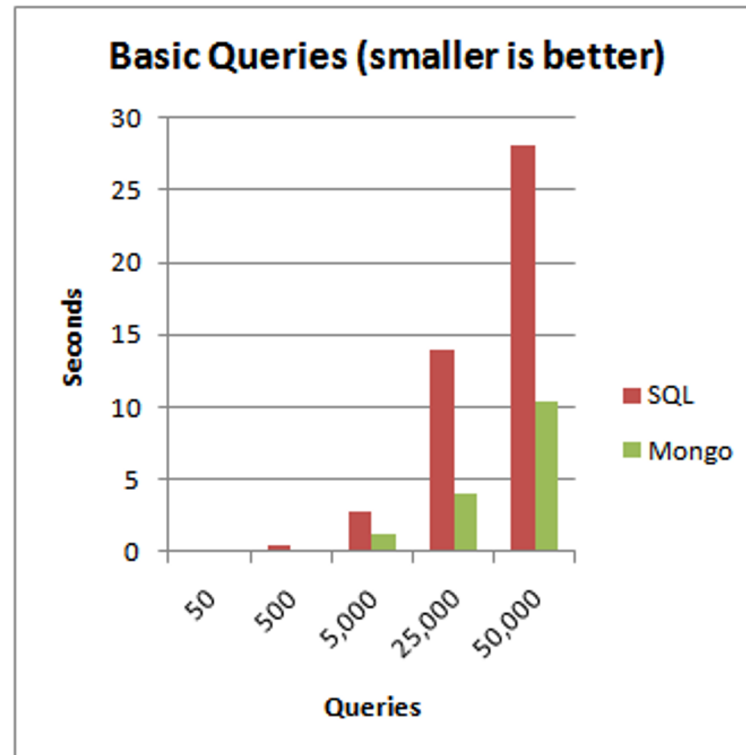
Support

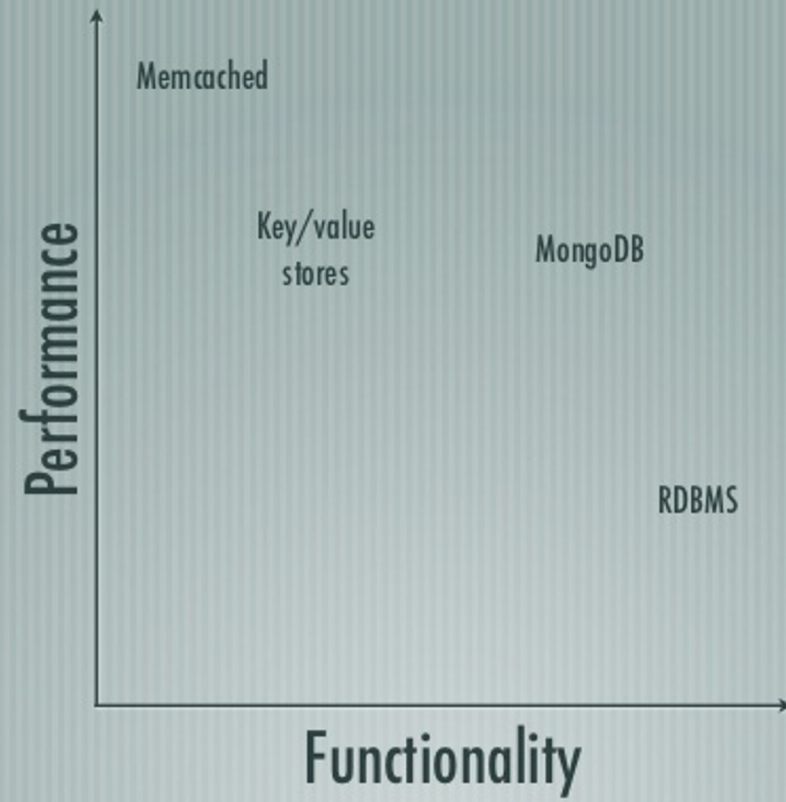
- 10Gen -> MongoDB Company
 - Sells enterprise version
 - Security and backup tools
 - Training
 - Integration
 - Support
 - Just raised \$150,000,000
 - Documentation - [MongoDB.org](https://www.mongodb.org)

Primary Benefits

- Speed Speed Speed!
- Rich Dynamic Queries
- Lazy Creation
- Schema-less
- Returns JSON
- Easy Replication and Failover
- Auto-Sharding
- MapReduce

How Fast?





A BSON (binary JSON) document

```
{
  "_id" : ObjectId("52832eb59f36fe144eeea8dc"),
  "baseprice" : 8.99,
  "category" : "toys",
  "colors" : [ "red", "green", "cosmic purple"],
  "name" : "Cosmic Yo-yo",
  "promotions" : [
    { "coupon" : "XY678", "saleprice" : 7.99, "expires" : ISODate("2013-12-12T00:00:00Z") },
    { "coupon" : "AB8888", "saleprice" : 7.49, "expires" : ISODate("2014-01-01T00:00:00Z") }
  ]
}
```

Terminology

Database -> Database

Table -> Collection

Record / Row -> Document

Field -> Field

find()

```
db.products.findOne()
```

```
db.products.find()
```

```
db.products.find().pretty()
```

```
db.products.find({ _id : ObjectId("52832eb59f36fe144eeea8dc")  
})
```

```
db.products.find({ name : "Cosmic Yo-yo" })
```

```
db.products.find({ name : /^hack/i }).pretty()
```

Index fields used for find!

Projections

```
db.products.find({ "name" : "Hacky Sack Maxx" }, { baseprice : 1 } )
```

```
db.products.find({ "name": "Hacky Sack Maxx" }, { baseprice: 1, category : 1 } )
```

```
db.products.find( { }, { promotions : 1 } )
```

```
db.products.find( { }, { promotions : 1 } )[0].promotions
```

Ids

ObjectId is a 12-byte **BSON** type, constructed using:

- a 4-byte value representing the seconds since the Unix epoch,
- a 3-byte machine identifier,
- a 2-byte process id, and
- a 3-byte counter, starting with a random value
- Show with `.getTimestamp()`

Queries

```
db.products.find().sort( { baseprice : -1  
}).pretty()
```

```
db.products.find().limit(1).pretty()
```

```
db.products.find().limit(1).skip(1).pretty()
```

Conditions

```
db.products.find({ baseprice : { $gt : 4.99 } })
```

```
db.products.find({ baseprice : { $lte : 4.99 } })
```

```
db.products.find({ promotions : { $lte : 4.99 } })
```

```
db.products.find( { colors : { $in : ["red"] } })
```

```
db.products.find( { "promotions.coupon" : { $in : [ "XY678"  
] } }).pretty()
```

```
db.products.find({ $and : [{ category : "toys"}, { baseprice : {  
$gt : 4.99 } }] })
```


Other Queries

```
db.products.count()
```

```
db.products.find({ baseprice : { $gt : 2.99 }}).count()
```

```
db.products.insert({ name : "Juggle-O-rama", baseprice : 11.99 })
```

```
db.products.update({ name : "Juggle-O-rama" }, { $set : { category : "toys" }})
```

```
db.products.update({ name: "Juggle-O-rama" }, { $set : { colors : ["silver",  
"gold"]}})
```

```
db.products.update({ name: "Juggle-O-rama" }, { $push : { colors : "sea foam  
green"}})
```

Don't forget \$set!

Aggregation

```
db.products.aggregate({ $group : { _id : "$category", totalprice : { $sum :  
"$baseprice" } } })
```

Primary Benefits

- Speed Speed Speed!
- Rich Dynamic Queries
- Lazy Creation
- Flexible Schema
- Returns JSON
- Easy Replication and Failover
- Auto-Sharding
- MapReduce

Lazy Creation

Lazy Creation Saves Developer Time

- Instant Set-up
- No Change Scripts
- Easier Data Migration
- Great for Data Warehousing

Flexible Schemas

- I.E. Different data for different product types
- Flexible nesting rules
- Simplifies Internationalization
- Easy Custom Fields

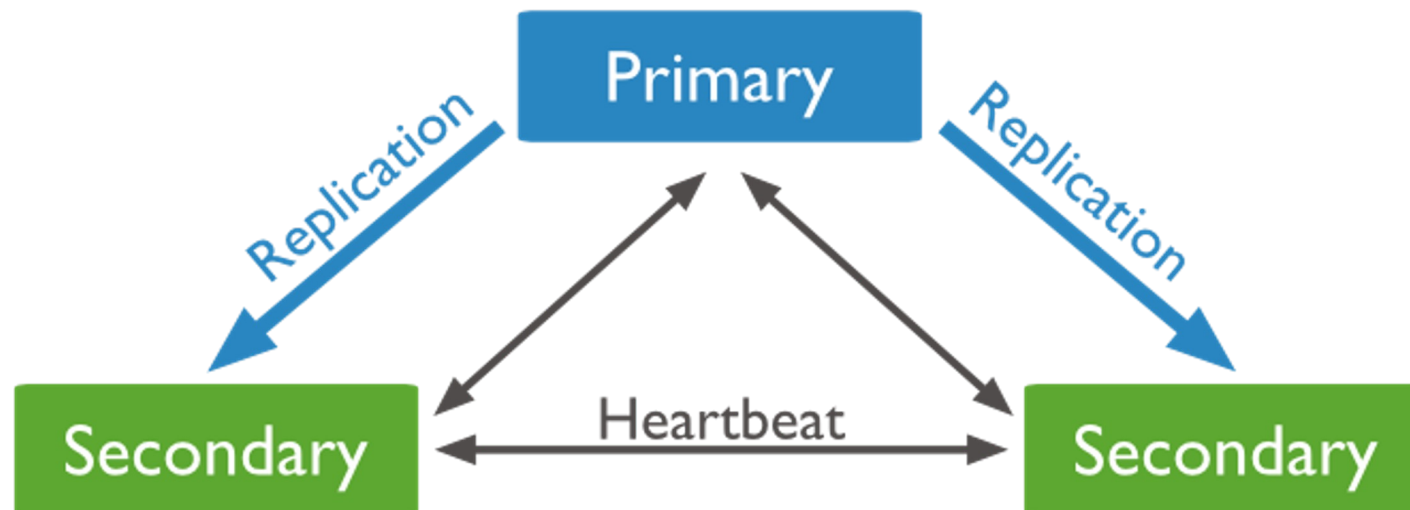
JSON Front-to-Back!

Queries return JSON

- Dirt simple APIs
- No Data Manipulation Needed
- Ditch the ORM
- Easy JavaScript Integration
- Fits perfectly with KnockoutJS, AngularJS etc.

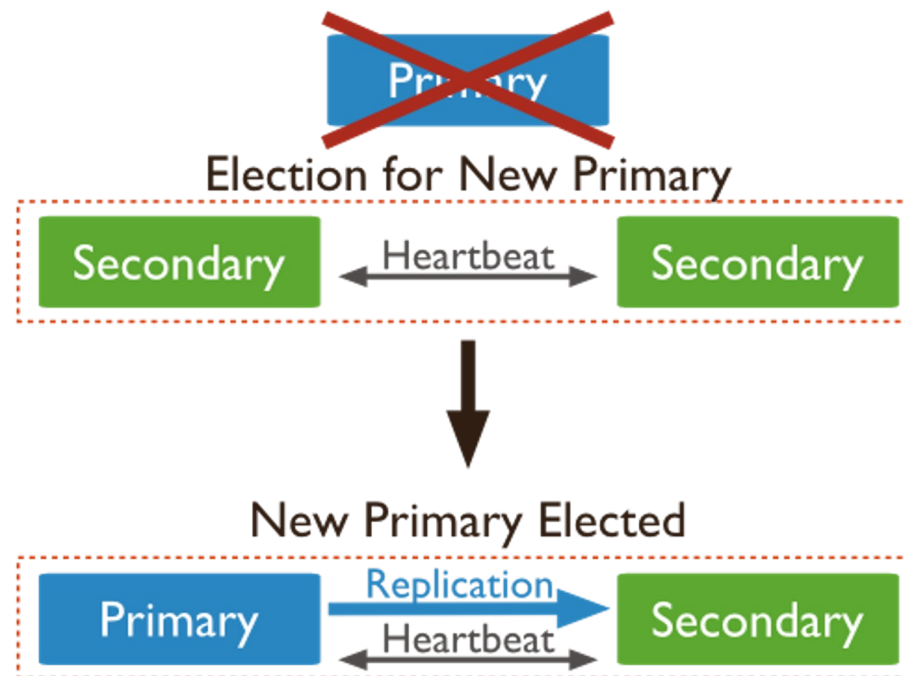
Replica Sets

Master / Slave

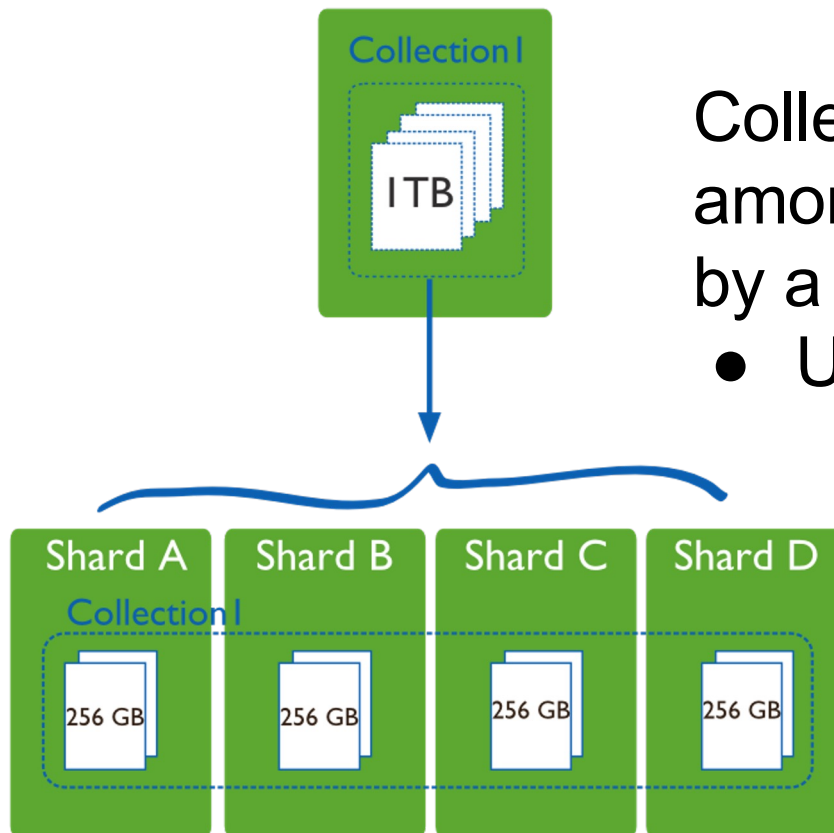


Replica Set Election

If a primary fails, the secondaries automatically elect a new primary



Sharding



Collections are split amongst multiple servers by a shard key

- Used to scale writes

Limitations

- No Transactions
- No Joins
- RAM intensive
- No referential integrity
- Eventual Consistency

Design Considerations

Embed vs. Reference

- Instead of joining junction tables, embed subdocuments in documents
- 90% of the time choose embed over reference
- You may have to store the same data twice

Denormalized Data

```
{  
  _id : ObjectId(...),  
  Name: "November Specials",  
  Promotions : [  
    { Title: "20% off all Yo-yos", Coupon: "AB345" },  
    { Title: "Free shipping on Hacky Sacks", Coupon : "XY456" }  
  ],  
  Dates : [ ISODate("2013-11-01"), ISODate("2013-11-31") ]  
}, {  
  _id : ObjectId(...),  
  Name: "December Specials",  
  Promotions : [  
    { Title: "10% off all frisbees", Coupon: "BA445" },  
    { Title: "Free overnight shipping on all jump ropes", Coupon : "XY456" }
```

Schema Design

SQL: Optimizing how data is stored

MongoDB: Optimize how data is used

SQL: What answers do I have?

MongoDB: What questions do I have?

Choices

```
{
  OrderId("....")
  Items : [
    { _id : ObjectId("..."), name : "Cosmic Yo-yo", color : "red", qty
: 1 },
    { _id : ObjectId("..."), name : Hackey Sack Maxx", color : "tiger
orange", qty: 2 }
  ]
  Promotions : [
    { _id : ObjectId("..."), Coupon : "AB456" : 6.99 }
  ]
}
```

Use Cases

- Anything with user generated data
 - Social Media
 - CMS
 - Blogs
- Product Data (Ecommerce)
- Games
- Location services
- Logging
 - Clickstream
- Analytics
 - Real-Time
 - Data Warehouses

Not Great For

Transaction Critical Data

- Purchases
- Banking
- Inventory Control

Use Both!

Products in MongoDB

```
{
  "_id" : ObjectId("52833435add826d9da8..."),
  "baseprice" : 4.99,
  "category" : "toys",
  "colors" : [ "tiger orange", " canary yellow" ],
  "name" : "Hacky Sack Maxx",
  "promotions" : [ { "coupon" : "ZY678", "sale_start_date" : "2006-11-28", "sale_end_date" : "2006-12-31" }, { "coupon" : "C1" } ]
}
```

Orders in SQL

