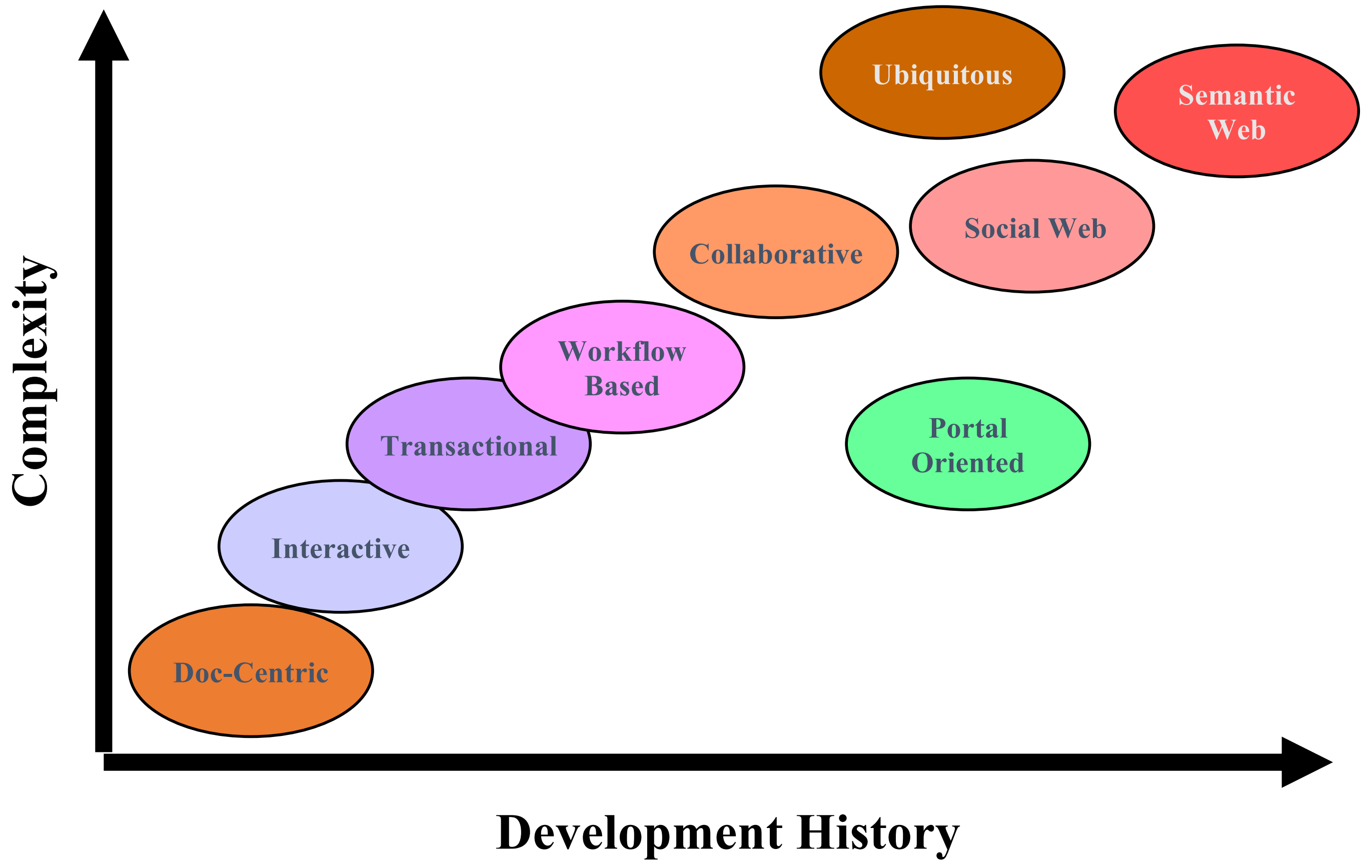
Web Database

- 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

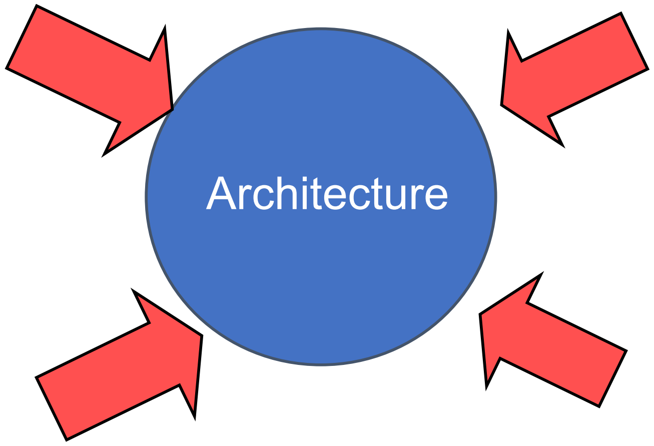


Architecture Defined

- 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



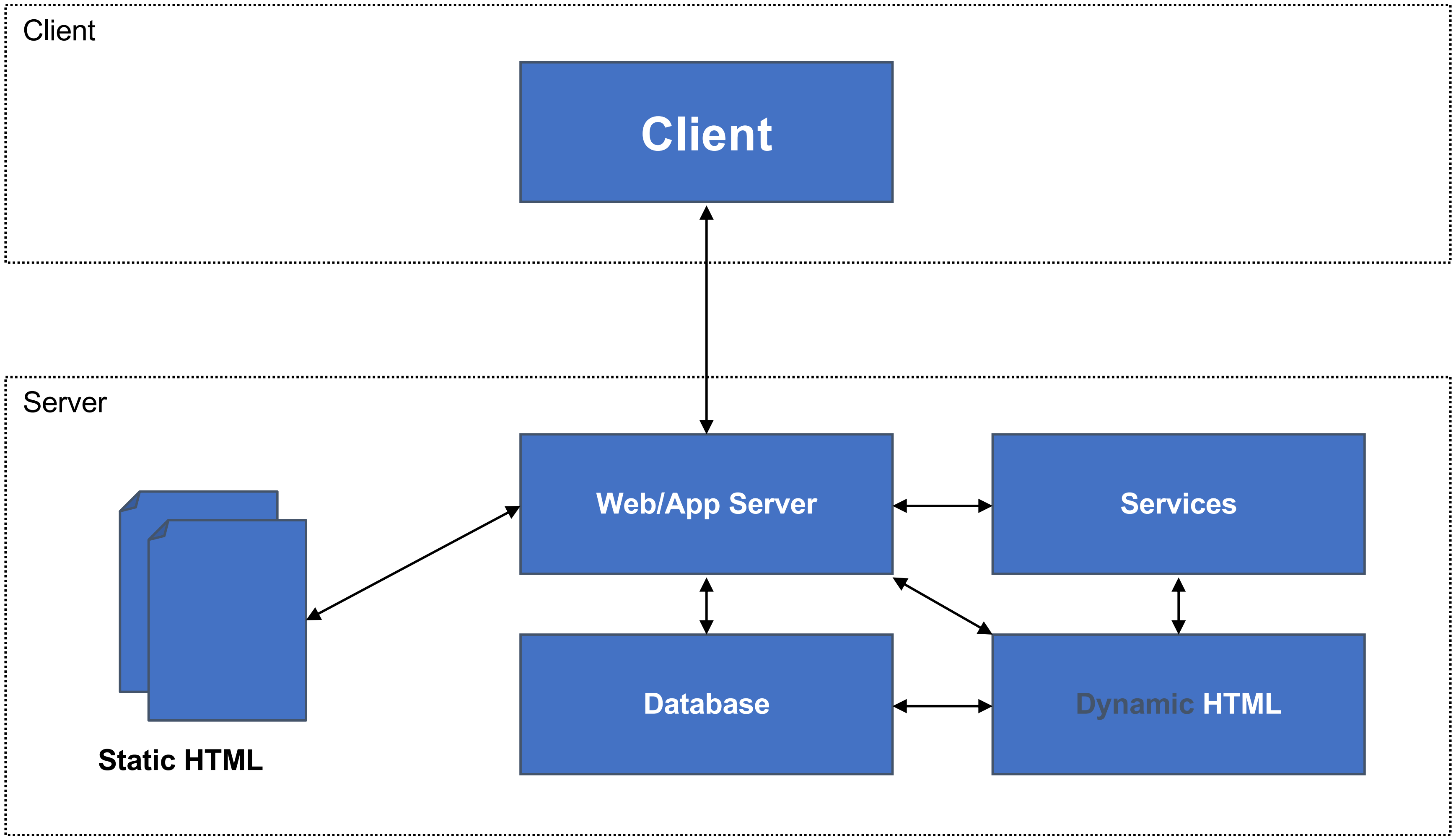
Functional Requirements
- Clients
- Users
- Other Stakeholders

Quality considerations with
- Performance
- Scalability
- Reusability
- Other?

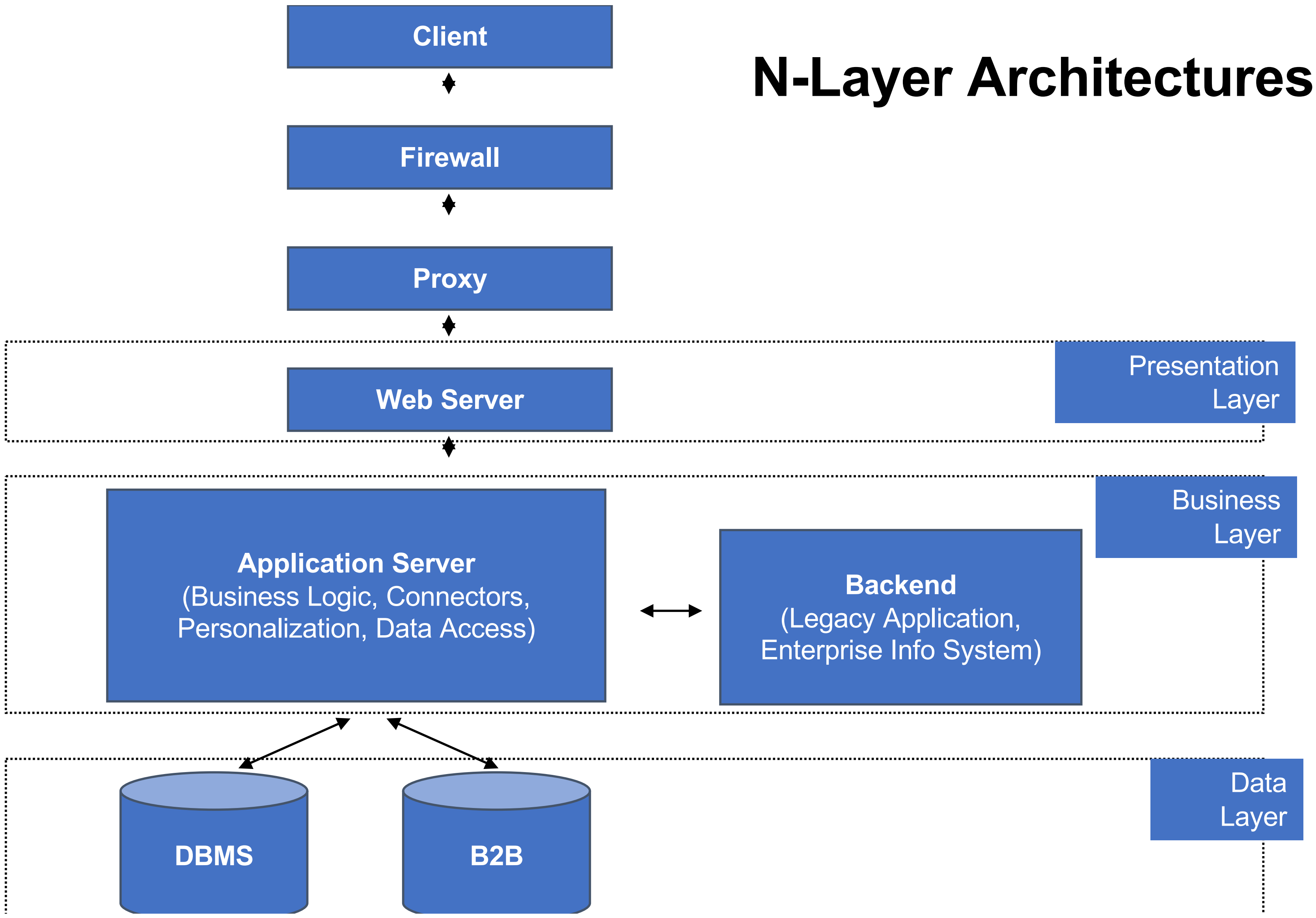
Experience with
- Existing Architecture
- Patterns
- Project Management
- Other?

Technical Aspects
- Operating System
- Middleware
- Legacy Systems
- Other?

Client/Server (2-Layer)



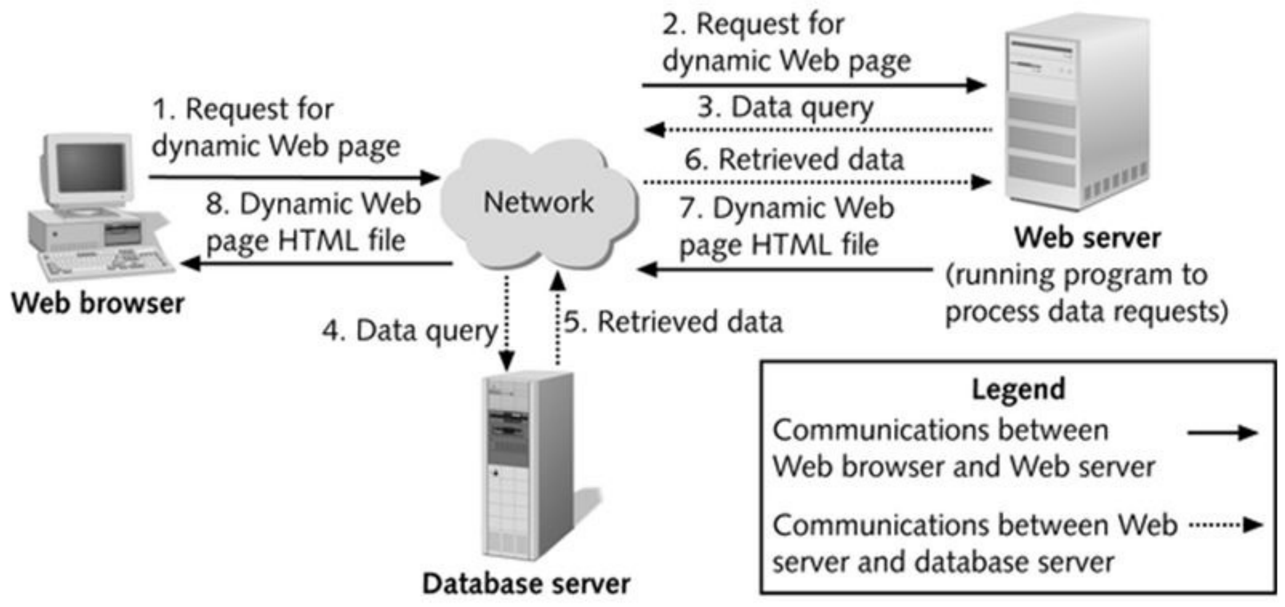
N-Layer Architectures



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.



1. Request for dynamic Web page
8. Dynamic Web page HTML file
Web browser
Network
2. Request for dynamic Web page
3. Data query
6. Retrieved data
7. Dynamic Web page HTML file
Web server
(running program to process data requests)
4. Data query
5. Retrieved data
Database server
Legend
Communications between Web browser and Web server
Communications between Web server and database server

Types of web databases

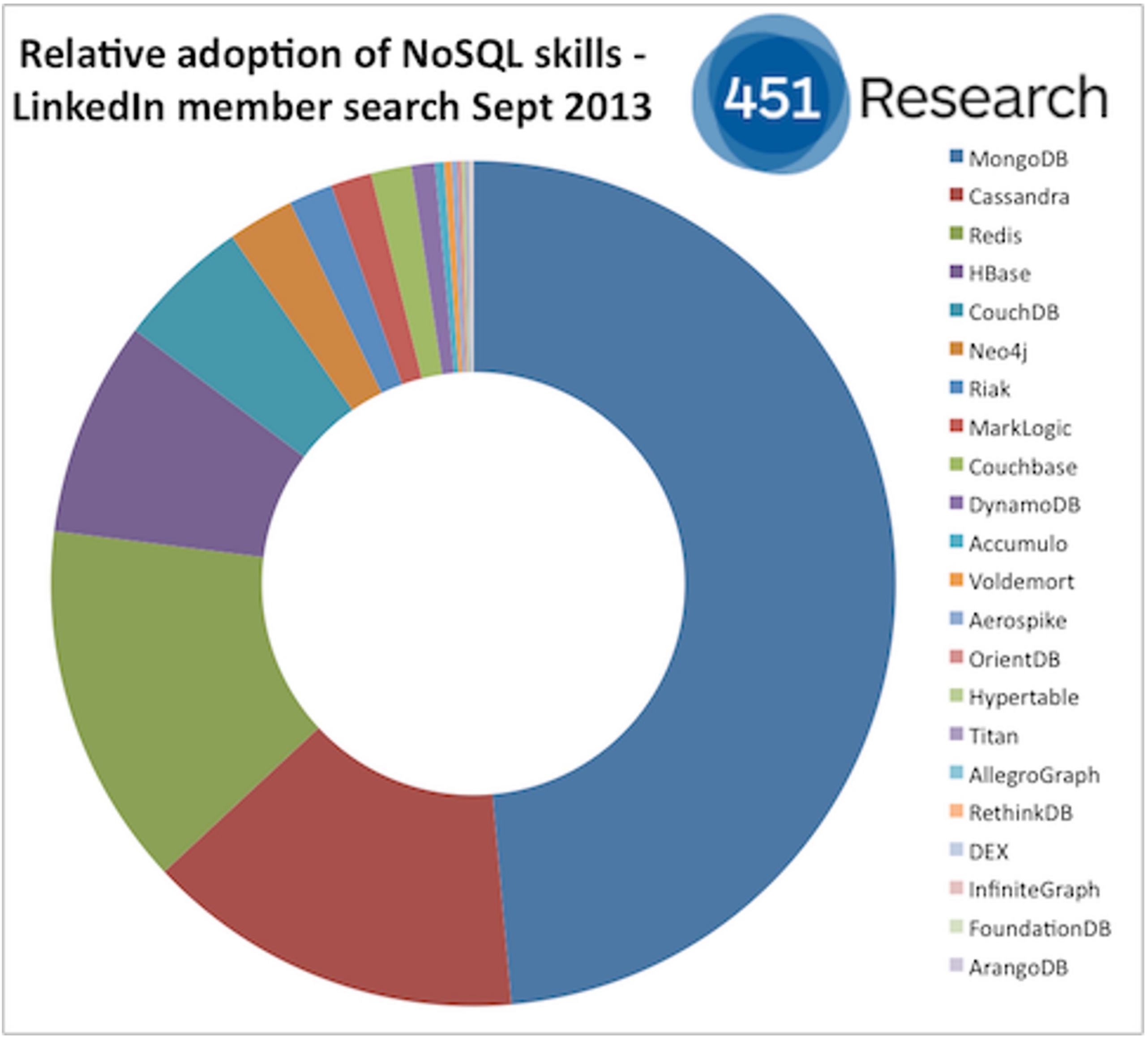
- MySQL - **open-source DBMS for relational databases.** – coupled with PHP (**LAMP** stack)
- PostgreSQL
- MongoDB - **stores data as collections of documents**

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



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

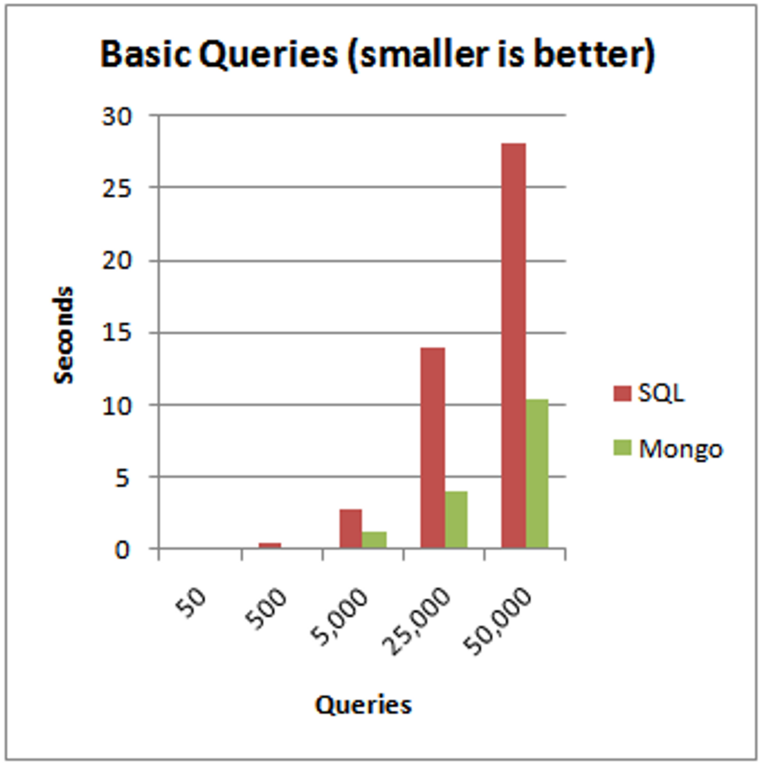
Support

- 10Gen -> MongoDB Company
    - Sells enterprise version
        - Security and backup tools
        - Training
        - Integration
        - Support
    - Just raised $150,000,000
    - Documention - [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?


Basic Queries (smaller is better)
Seconds
30
25
20
15
10
5
0
50
500
5,000
25,000
50,000
Queries
🟥 SQL
🟩 Mongo

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

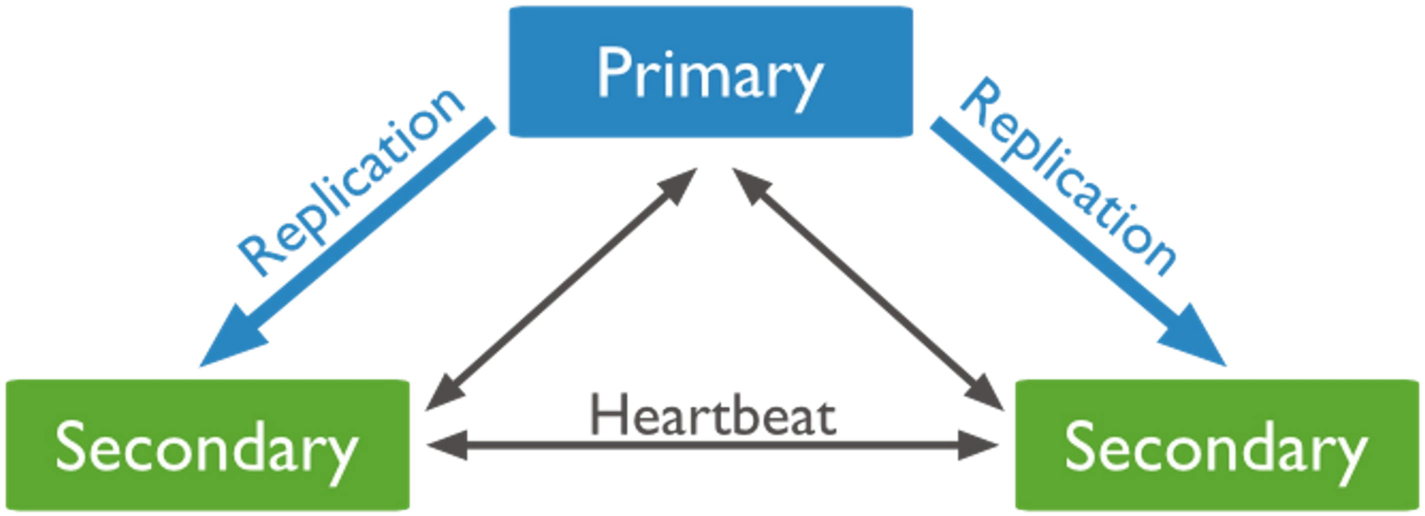
JSON Front-to-Back!

Queries return JSON

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

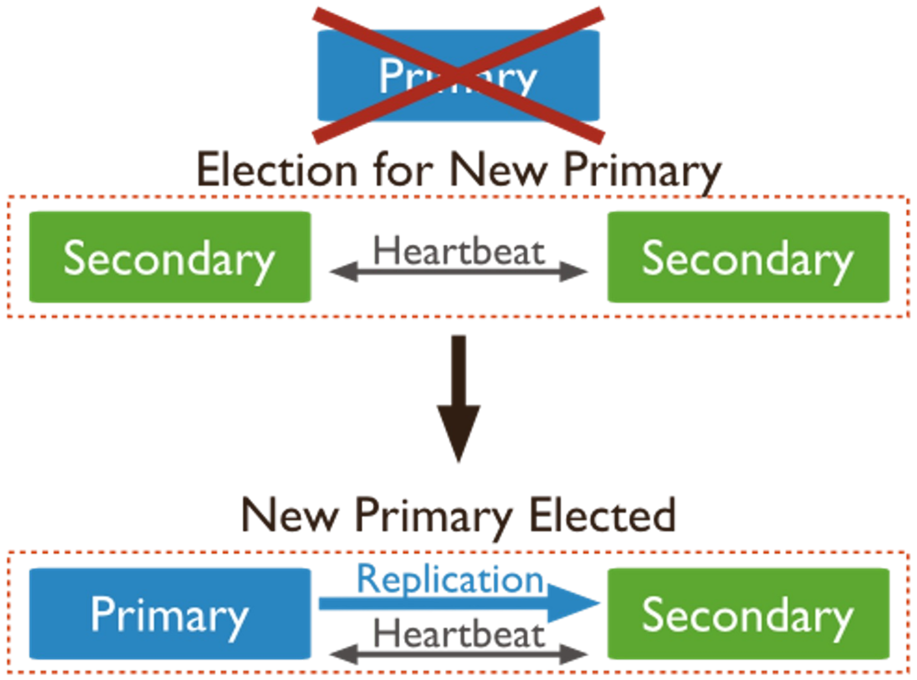
Replica Sets

Master / Slave

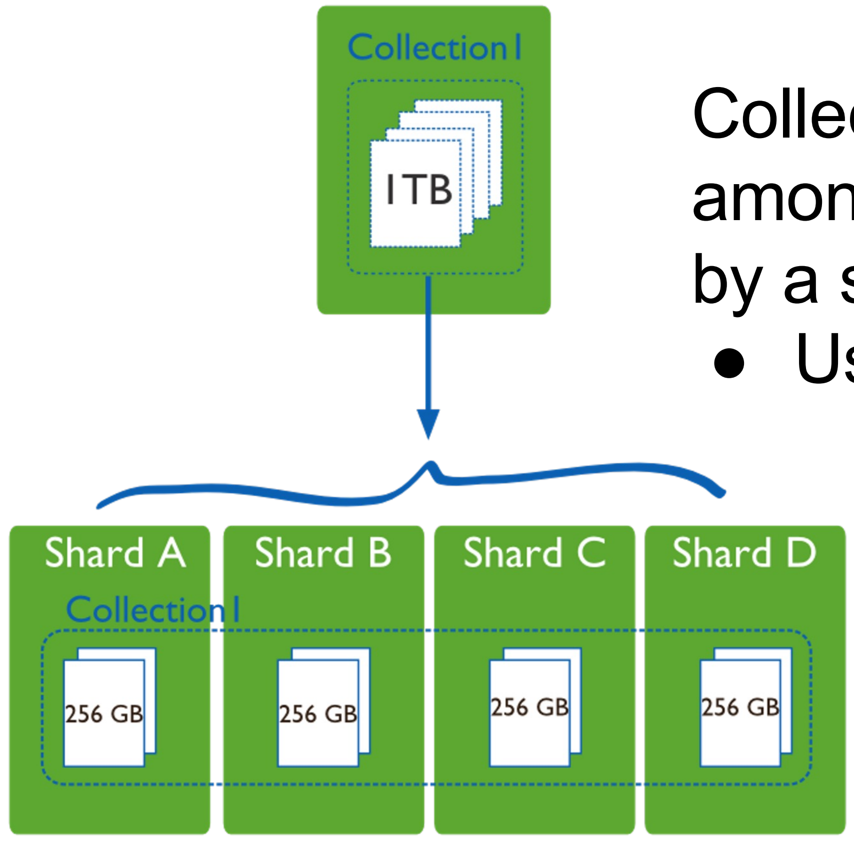


Replica Set Election

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?

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

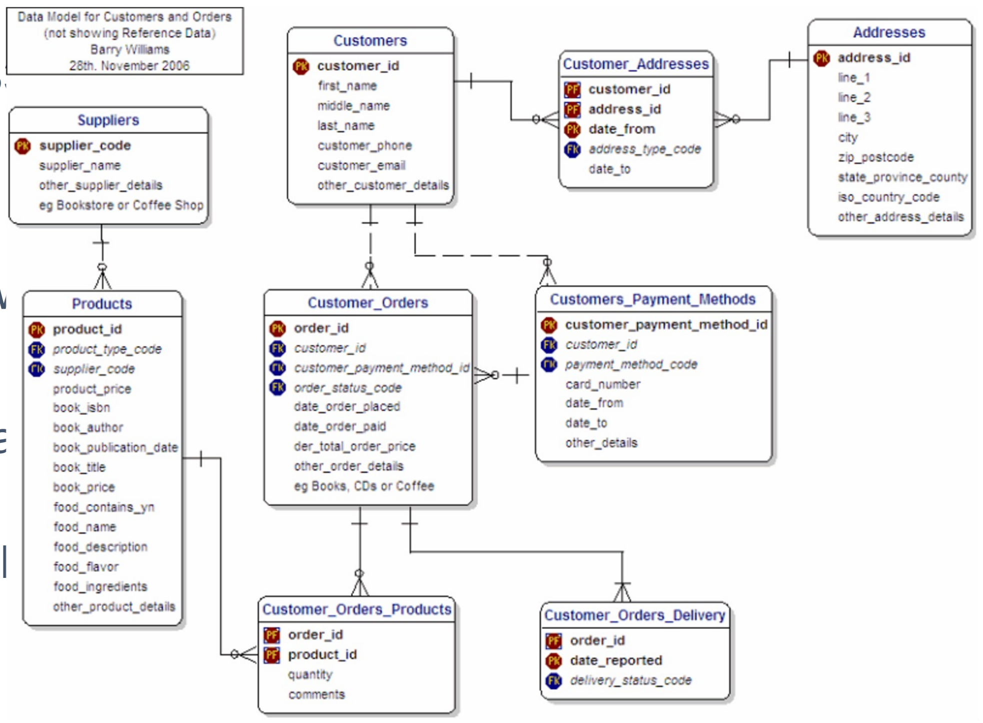
- Purchases
- Banking
- Inventory Control

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", "sa
},
                              { "coupon" : "Cl
                          ]
}
```

Orders in SQL



Data Model for Customers and Orders (not showing Reference Data) Barry Williams 28th. November 2006
Suppliers
PK supplier_code
supplier_name
other_supplier_details
eg Bookstore or Coffee Shop
Customers
PK customer_id
first_name
middle_name
last_name
customer_phone
customer_email
other_customer_details
Customer_Addresses
PF customer_id
PF address_id
PK date_from
FK address_type_code
date_to
Addresses
PK address_id
line_1
line_2
line_3
city
zip_postcode
state_province_county
iso_country_code
other_address_details
Products
PK product_id
FK product_type_code
FK supplier_code
product_price
book_isbn
book_author
book_publication_date
book_title
book_price
food_contains_yn
food_name
food_description
food_flavor
food_ingredients
other_product_details
Customer_Orders
PK order_id
FK customer_id
FK customer_payment_method_id
FK order_status_code
date_order_placed
date_order_paid
der_total_order_price
other_order_details
eg Books, CDs or Coffee
Customers_Payment_Methods
PK customer_payment_method_id
FK customer_id
FK payment_method_code
card_number
date_from
date_to
other_details
Customer_Orders_Products
PF order_id
PF product_id
quantity
comments
Customer_Orders_Delivery
PF order_id
PK date_reported
FK delivery_status_code