MongoDB – An Overview

21-Oct-2016

Socrates

Agenda

- What is NoSQL DB?
- Types of NoSQL DBs
- DBMS and MongoDB Comparison
- Why MongoDB?
- MongoDB Architecture
  - Storage Engines
  - Data Model
  - Query Language
  - Security
  - Data Management
- Query Language in Detail …

What is NoSQL DB?

- Not a Replacement of SQL
- Not a Traditional RDBMS
- Does not support ACID property
- Does not built on Tables
- Not a Silver-bullet solution

- Schema-less or Dynamic Schema
- Highly Distributed (mostly built-in)
- High Performance
- Rich Query Language
- High Availability (due to Replication)
- High Scalable

Types of NoSQL DBs

Key-Value Store	Example
• Basic and Simplest form • Stored as Key-Value pair	• Riak • Redis

Document Store	Example
• Stored as Document • Document may have different fields	• MongoDB • CouchDB

Column-Family Store	Example
• Each column stored in separate file • Automatic Vertical partitioning • Improved compression	• Cassandra • HBase

Graph Store	Example
• Simpler and more expressive • Based on Node and Relationship	• Neo4J • Giraph

DBMS and MongoDB Comparison

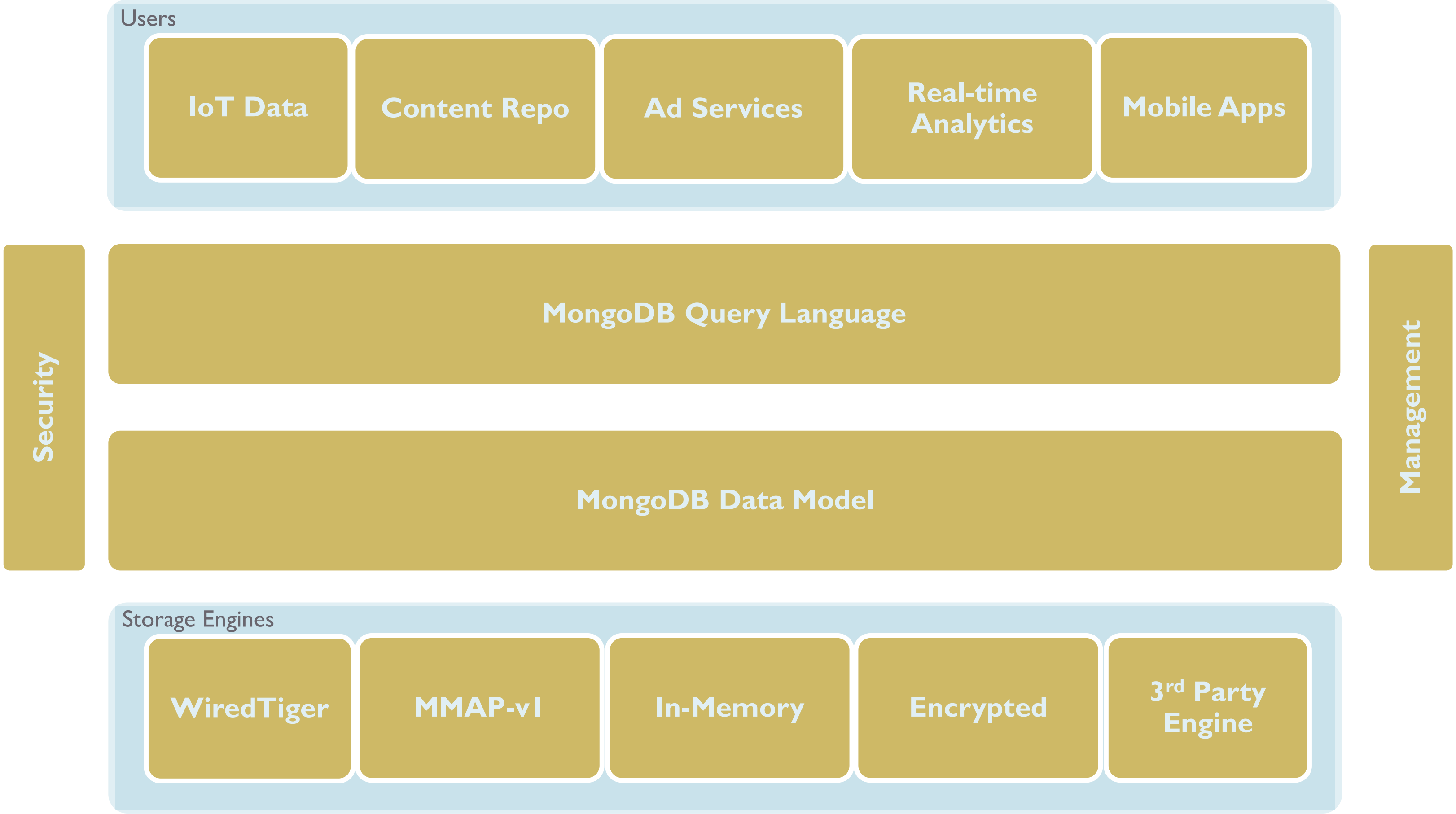
DBMS Terms/ Concepts	MongoDB Terms/ Concepts
Database	Database
Table	Collection
Row	Document
Column	Field
Stored as Defined datatype	Stored as BSON (Binary JSON*)

* JSON – Javascript Object Notation

Why MongoDB?

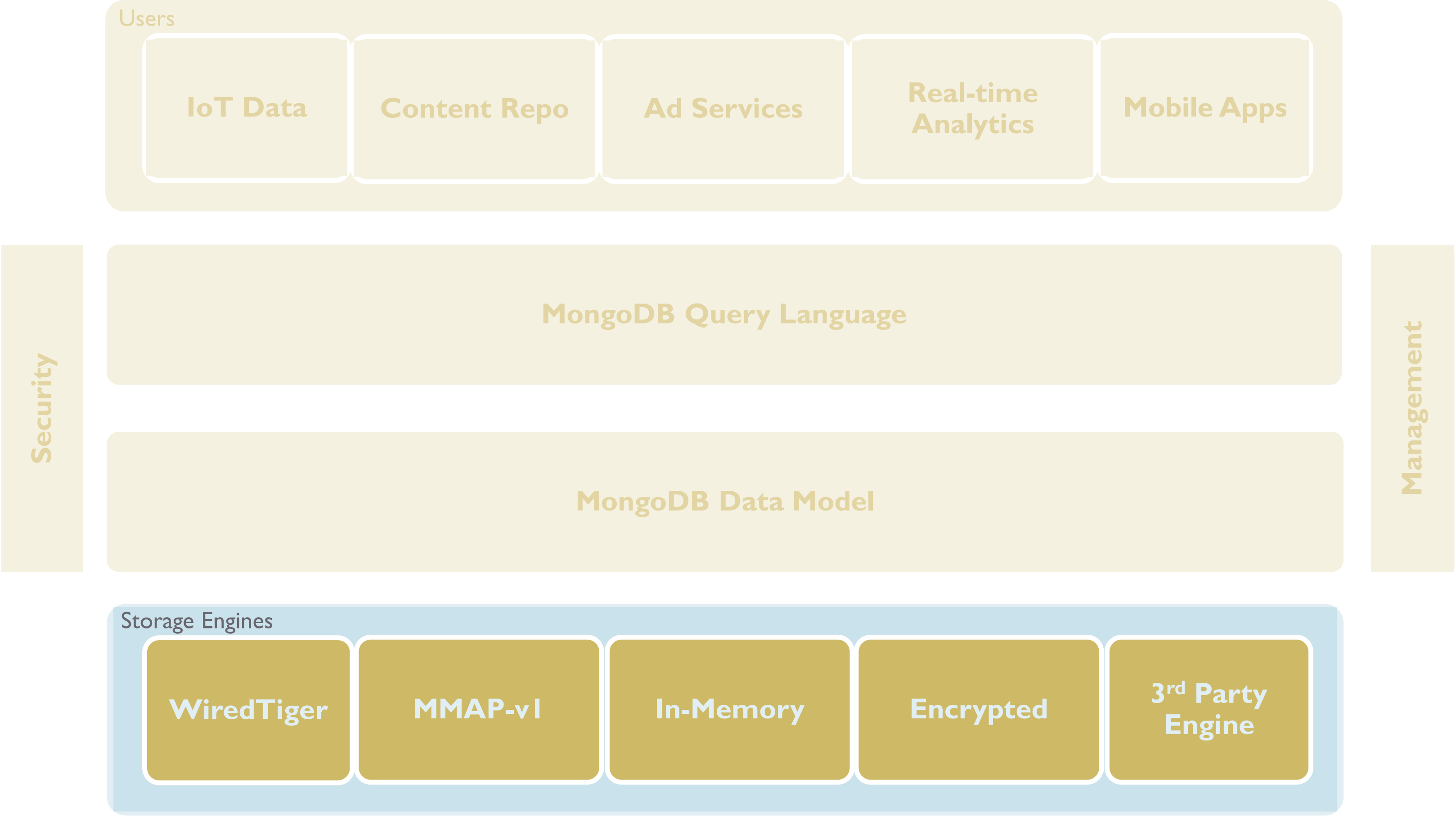
- Document Oriented (Rich Documents)
- Strong Consistency
- Built-in Sharding (Distributed)
- Data stored in BSON format
- Automatic Data Deletion (using TTL index)
- No pre-Table (Collection) creation
- No Schema (Schema-less)
- No Joins; but pre-Join and Embedded data
- No Constraints – (say Foreign-Key)
- No Transaction
- Document size upto16MB (16,777,216 bytes)

MongoDB Architecture



Nexus Architecture combining NoSQL and RDBMS

MongoDB Architecture



Users

IoT Data

Content Repo

Ad Services

Real-time Analytics

Mobile Apps

Security

MongoDB Query Language

MongoDB Data Model

Management

Storage Engines

WiredTiger

MMAP-v1

In-Memory

Encrypted

3rd Party Engine

Storage Engines

Memory MAP (MMAP)

- Collection-level Locking
- *Write-ahead* Journal Log for Data Recovery
- No Data Compression

WiredTiger

- Document-level Locking
- *Write-later* Journal Log
- *Snappy* and *ZLib* Data Compression

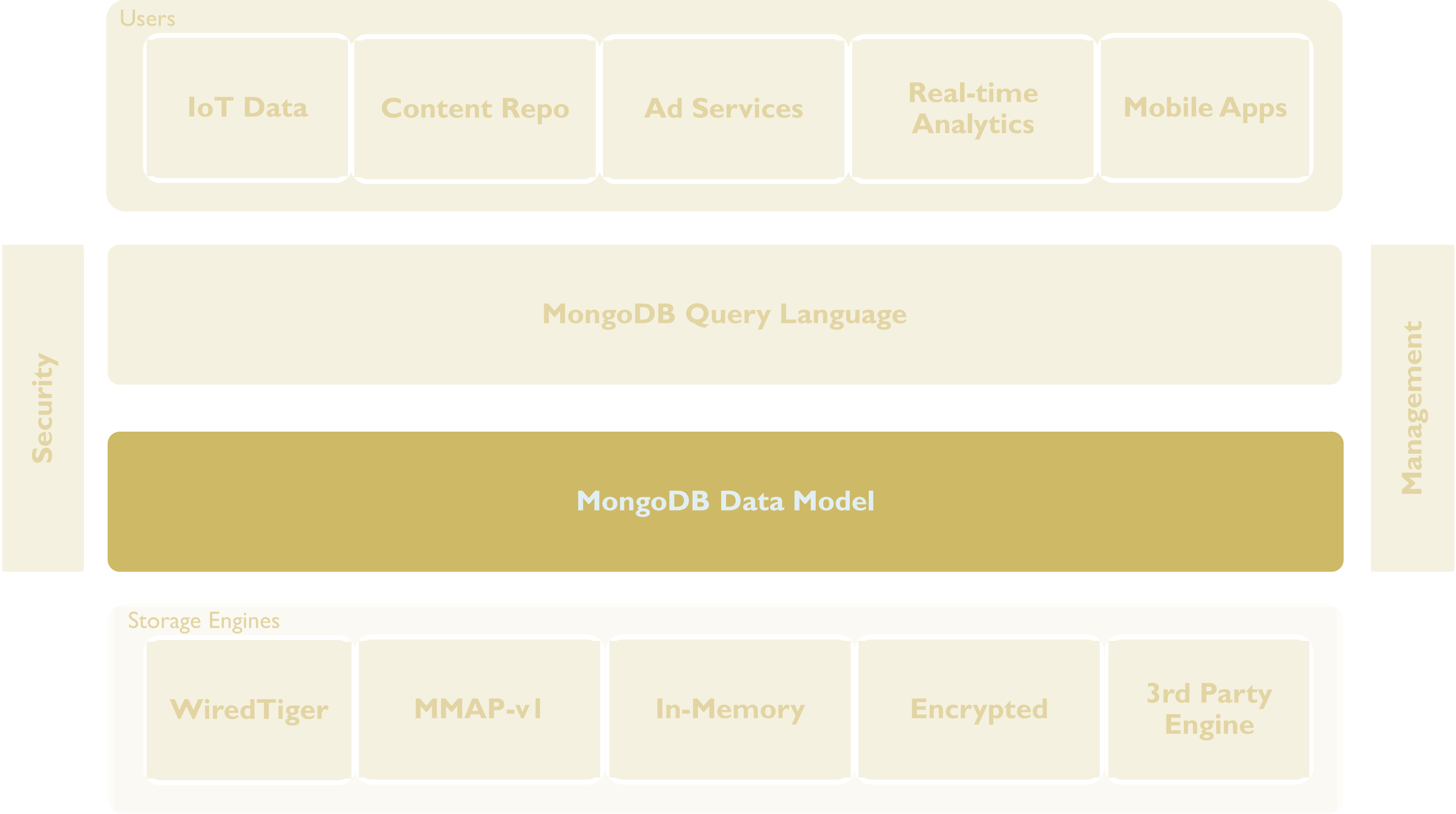
In-Memory

- Does not maintain data on Disk
- Used by high-performance real-time analytical apps

Encrypted

- Optional Encryption on top of WiredTiger
- Encrypted Data stored in File System
- Unencrypted Data in memory and in-flight

MongoDB Architecture



Users

IoT Data

Content Repo

Ad Services

Real-time Analytics

Mobile Apps

Security

MongoDB Query Language

MongoDB Data Model

Management

Storage Engines

WiredTiger

MMAP-v1

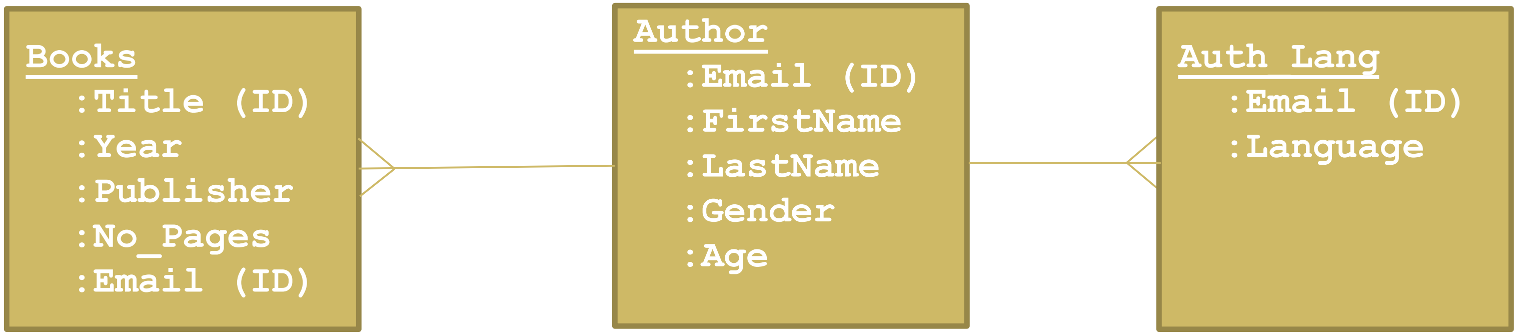
In-Memory

Encrypted

3rd Party Engine

Data Model Comparison

RDBMS Data Model



Books
:Title (ID)
:Year
:Publisher
:No_Pages
:Email (ID)

Author
:Email (ID)
:FirstName
:LastName
:Gender
:Age

Auth_Lang
:Email (ID)
:Language

AUTHOR				
Email	FirstName	LastName	Gender	Age
bob@gmail.com	Bob	Johnson	M	30

AUTH_LANG	
Email	Language
bob@gmail.com	English
bob@gmail.com	Spanish
bob@gmail.com	German

BOOKS				
Title	Year	Publisher	No_Pages	**Email**
Learn MongoDB in 30 days	2013	O'Reilly Publications	284	bob@gmail.com
MongoDB – Tips and Tricks	2015	O'Reilly Publications	367	bob@gmail.com
MongoDB for Dummies	2014	McGraw-Hill Publications	148	bob@gmail.com

Data Model Comparison

MongoDB Data Model

```
Author
   :FirstName
   :LastName
   :Gender
   :Age
   :Email
```

```
{    '_id'       : 1,
     'FirstName' : 'Bob',
     'LastName'  : 'Johnson',
     'Gender'    : 'M',
     'Age'       : '30',
     'Email'     : 'bob@gmail.com'
}
```

Data Model Comparison

MongoDB Data Model

```
Author
   :FirstName
   :LastName
   :Gender
   :Age
   :Email

Books []
   :Title
   :Year
   :Publisher
   :No_Pages
```

```
{   '_id'       : 1,
    'FirstName' : 'Bob',
    'LastName'  : 'Johnson',
    'Gender'    : 'M',
    'Age'       : '30',
    'Email'     : 'bob@gmail.com',
    'Books'     : [
        {
         'Title': 'Learn MongoDB in 30 days',
         'Year' : 2013,
         'Publisher' : 'O’Reilly Publications',
         'No_Pages' : 284
        }, {
         'Title': 'MongoDB – Tips and Tricks',
         'Year' : 2015,
         'Publisher' : 'O’Reilly Publications',
         'No_Pages'  : 367
        }, {
         'Title': 'MongoDB for Dummies',
         'Year' : 2014,
         'Publisher': 'McGraw-Hill Publications',
         'No_Pages' : 148
        }
    ]
 }
```

Data Model Comparison

MongoDB Data Model

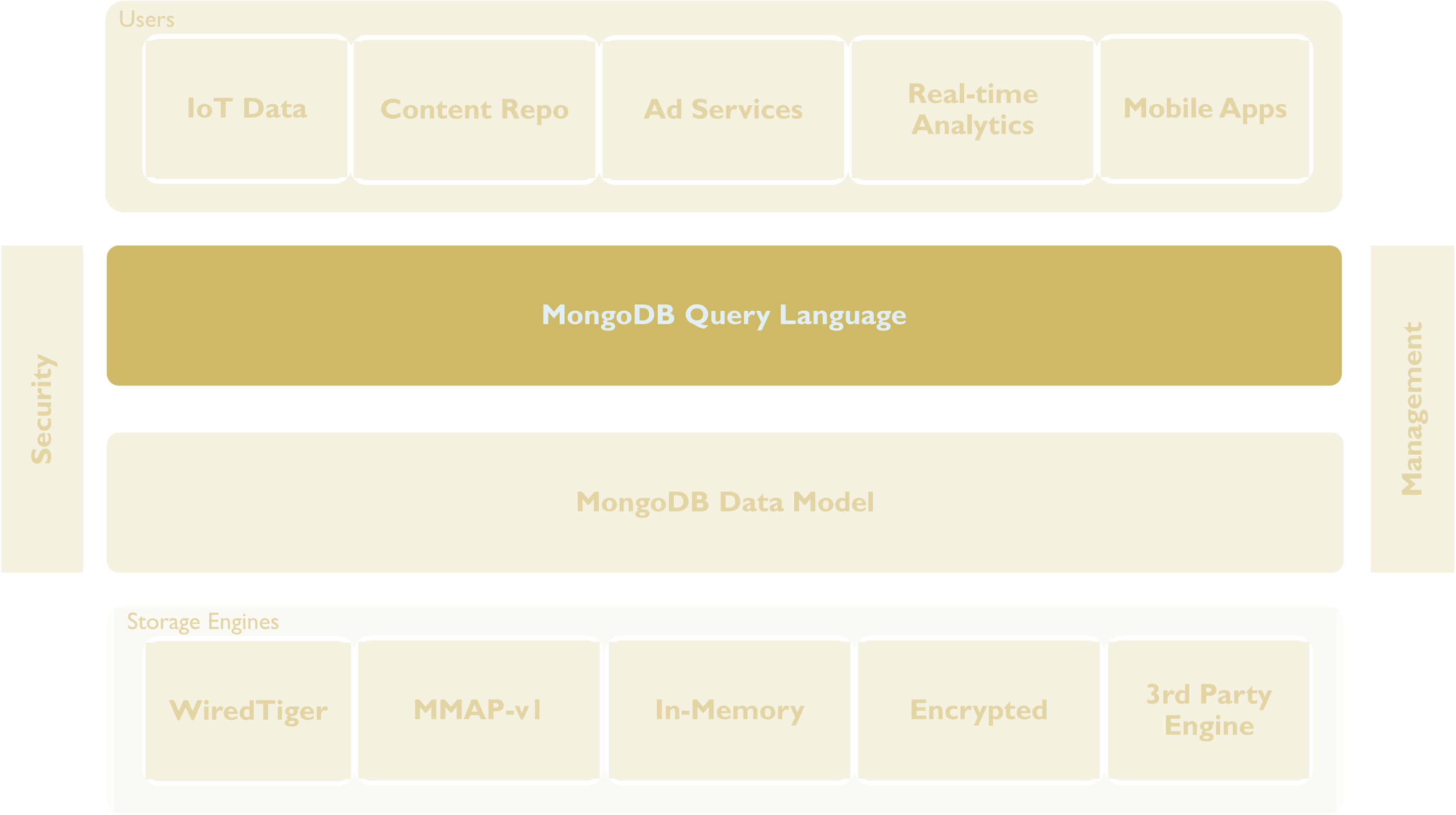
```
Author
   :FirstName
   :LastName
   :Gender
   :Age
   :Email

Books []
   :Title
   :Year
   :Publisher
   :No_Pages

Language []
    :Language
```

```
{   '_id'       : 1,
    'FirstName' : 'Bob',
    'LastName'  : 'Johnson',
    'Gender'    : 'M',
    'Age'       : '30',
    'Email'     : 'bob@gmail.com',
    'Books'     : [
        {
         'Title': 'Learn MongoDB in 30 days',
         'Year' : 2013,
         'Publisher' : 'O’Reilly Publications',
         'No_Pages' : 284
        }, {
         'Title': 'MongoDB – Tips and Tricks',
         'Year' : 2015,
         'Publisher' : 'O’Reilly Publications',
         'No_Pages'  : 367
        }, {
         'Title': 'MongoDB for Dummies',
         'Year' : 2014,
         'Publisher': 'McGraw-Hill Publications',
         'No_Pages' : 148
        }
    ],
  'Language' : ['English', 'Spanish', 'German']
}
```

MongoDB Architecture



Query Language

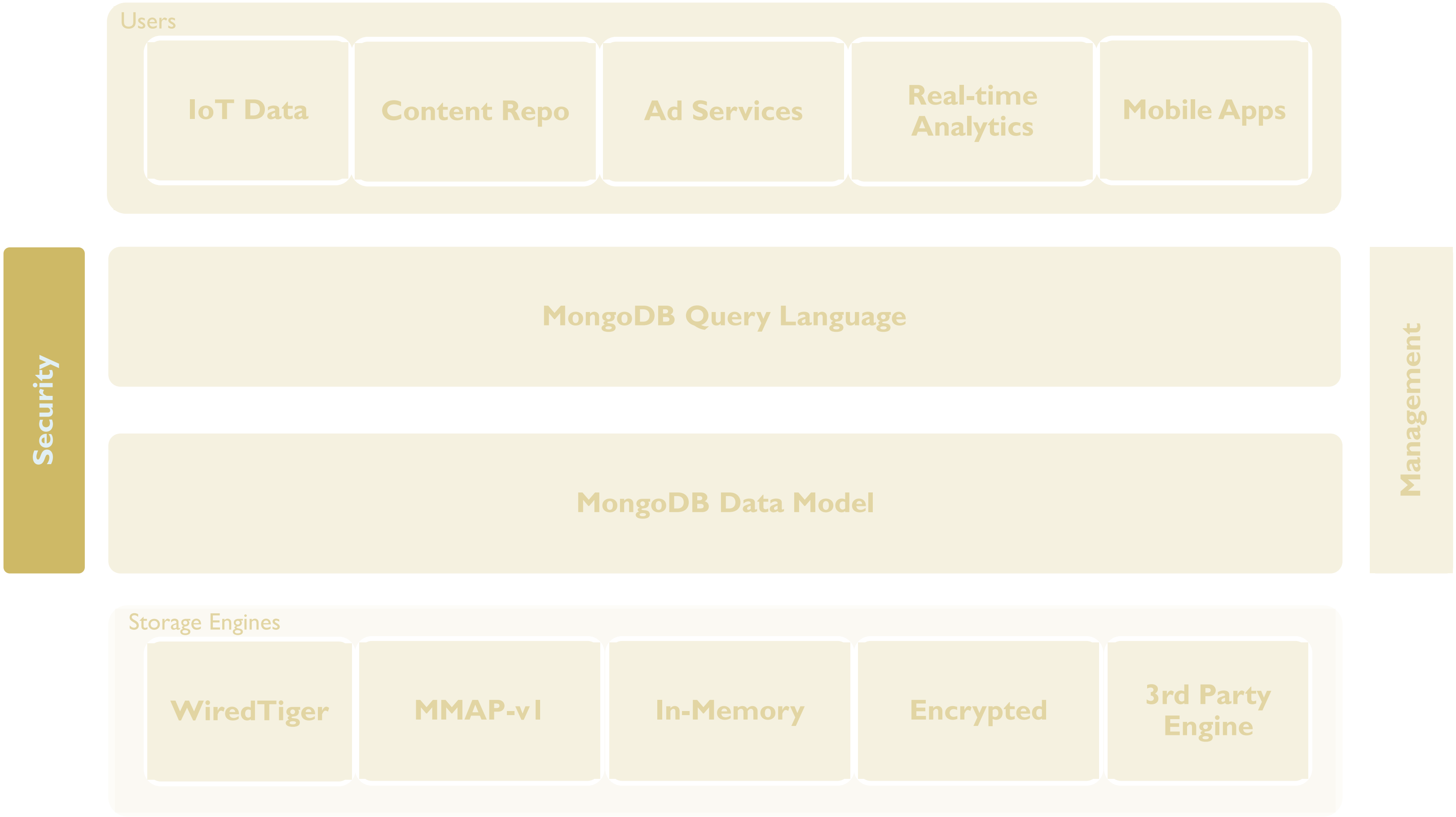
- Idiomatic Drivers
  - PHP, Java, Scala, Ruby, Python, PERL, .NET, JavaScript
- Interactive JavaScript Shell
- Interactive GUI – MongoDB Compass
- Simple to most Complex Queries and Data Visualization

Key-Value Query	Based on Keys
Range Query	Based on values (*between, less than, equal to…*)
Geospatial Query	Based on Longitude and Latitude (coordinates)
Text Search	Based on Text Arguments (*AND, OR, NOT*)
Aggregation Framework	Based on Numeric Values (*Count, Min, Max, Avg*)
Map-Reduce Query	Based on Data Processing needs (Complex)
Connector for Business Intelligence	

Query Language Contd..

- Indexing
  - Unique Indexes - On Single Field
  - Compound Indexes - On Multiple Fields
  - Array Indexes - On Field that contains Arrays
  - TTL Indexes - On Date field with Time-To-Live Seconds
  - Geospatial Indexes - On Geo-coordinates for 2 dimensional queries
  - Partial Indexes - On Field(s) with Filter condition
  - Sparse Indexes - On Field(s) that contain values (Null fields ignored)
  - Text Search Indexes - Specialized index (Stemming, stop-words etc)
- Query Optimization
  - Automatic selection query based on
    - Best Index
    - Predicates
    - Sort Criteria
- Covered Queries Results based on *Index Page* NOT from *Document*

MongoDB Architecture



Users

IoT Data

Content Repo

Ad Services

Real-time Analytics

Mobile Apps

Security

MongoDB Query Language

MongoDB Data Model

Management

Storage Engines

WiredTiger

MMAP-v1

In-Memory

Encrypted

3rd Party Engine

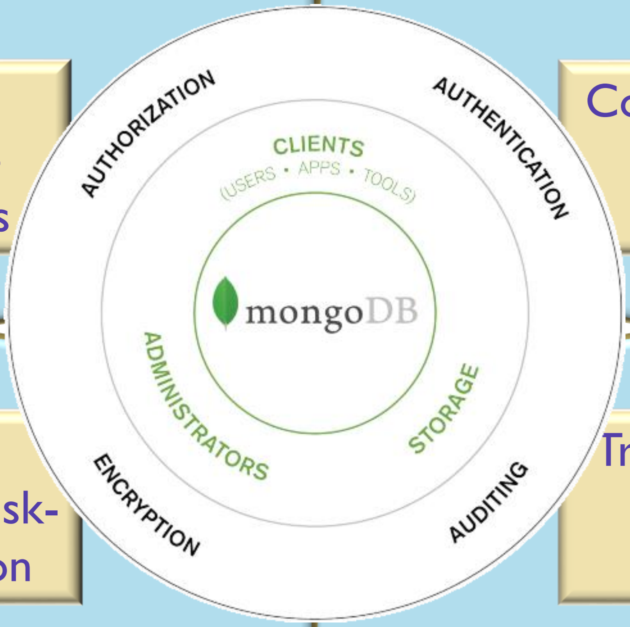
Security

Document and Field level permissions using *User-Defined roles*

Controls Permissions to Users and Apps

Using LDAP, Kerberos, Windows Active Directory

Controls Access to DB



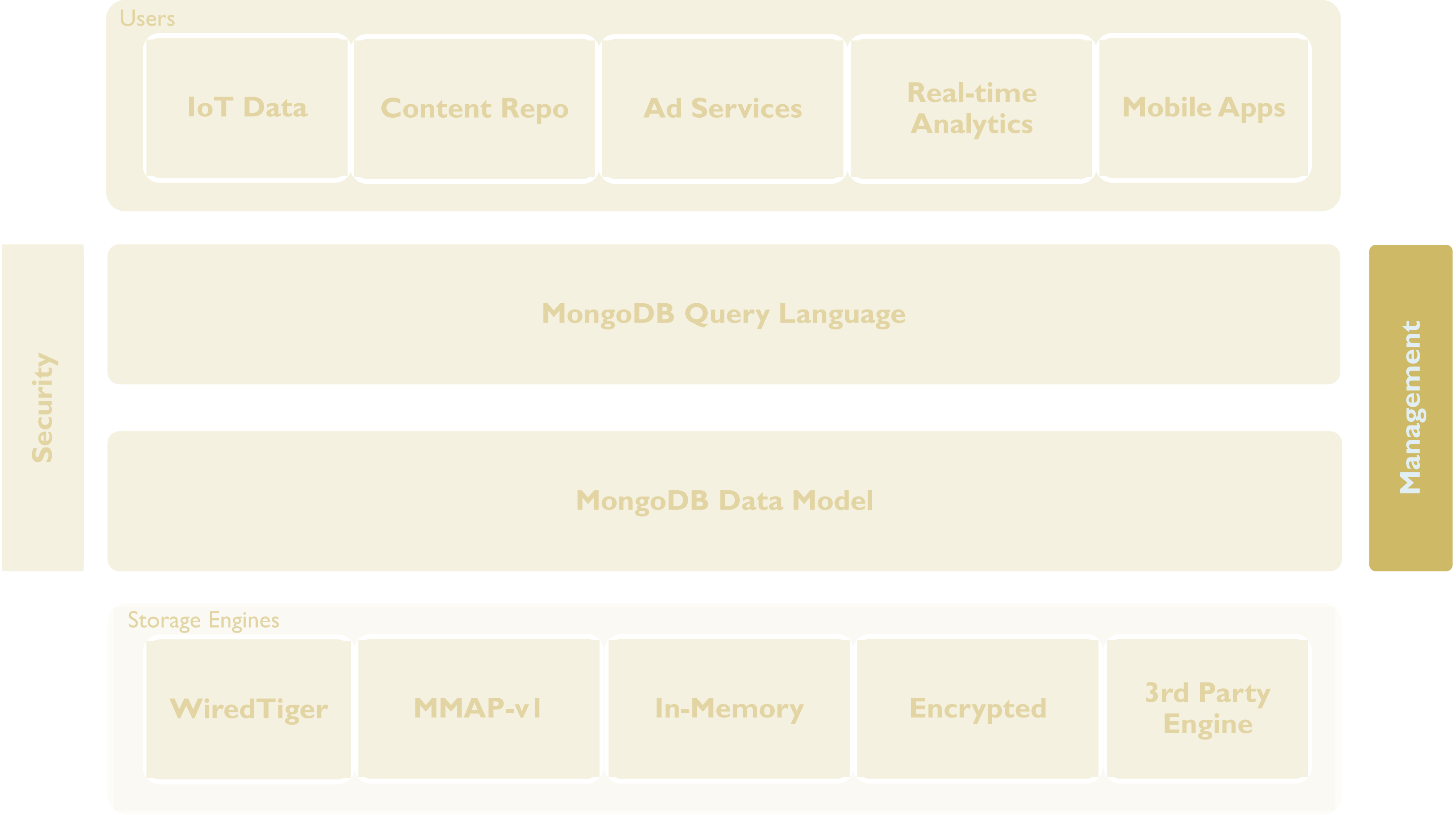
Network and Disk-Level Encryption

Network Level: Using SSL/TLS
Disk Level: Using Encrypted Storage Engine

Tracks DML, DCL, and DDL DB operations

Using native audit log

MongoDB Architecture



Users

IoT Data

Content Repo

Ad Services

Real-time Analytics

Mobile Apps

Security

MongoDB Query Language

MongoDB Data Model

Management

Storage Engines

WiredTiger

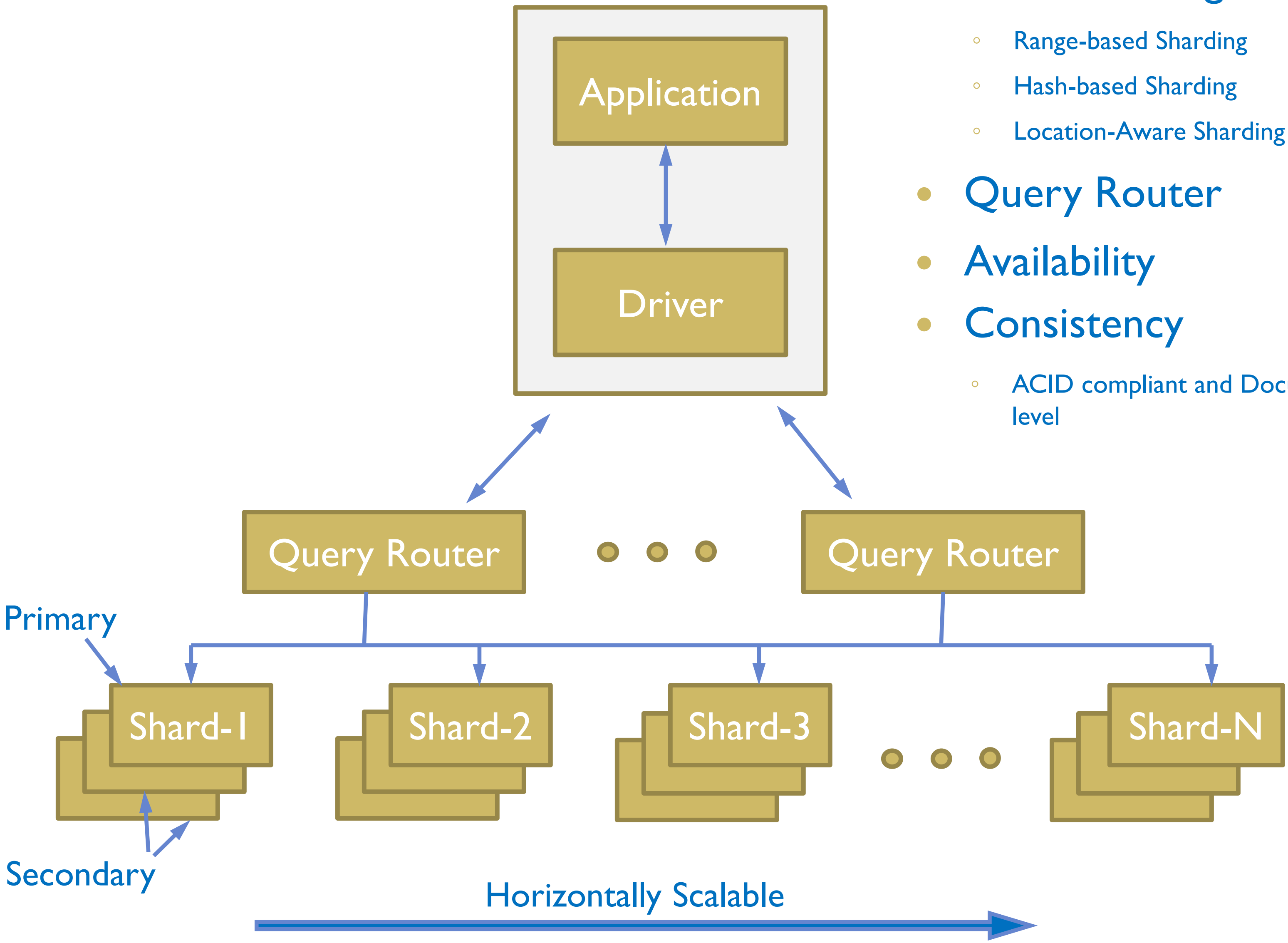
MMAP-v1

In-Memory

Encrypted

3rd Party Engine

Data Management

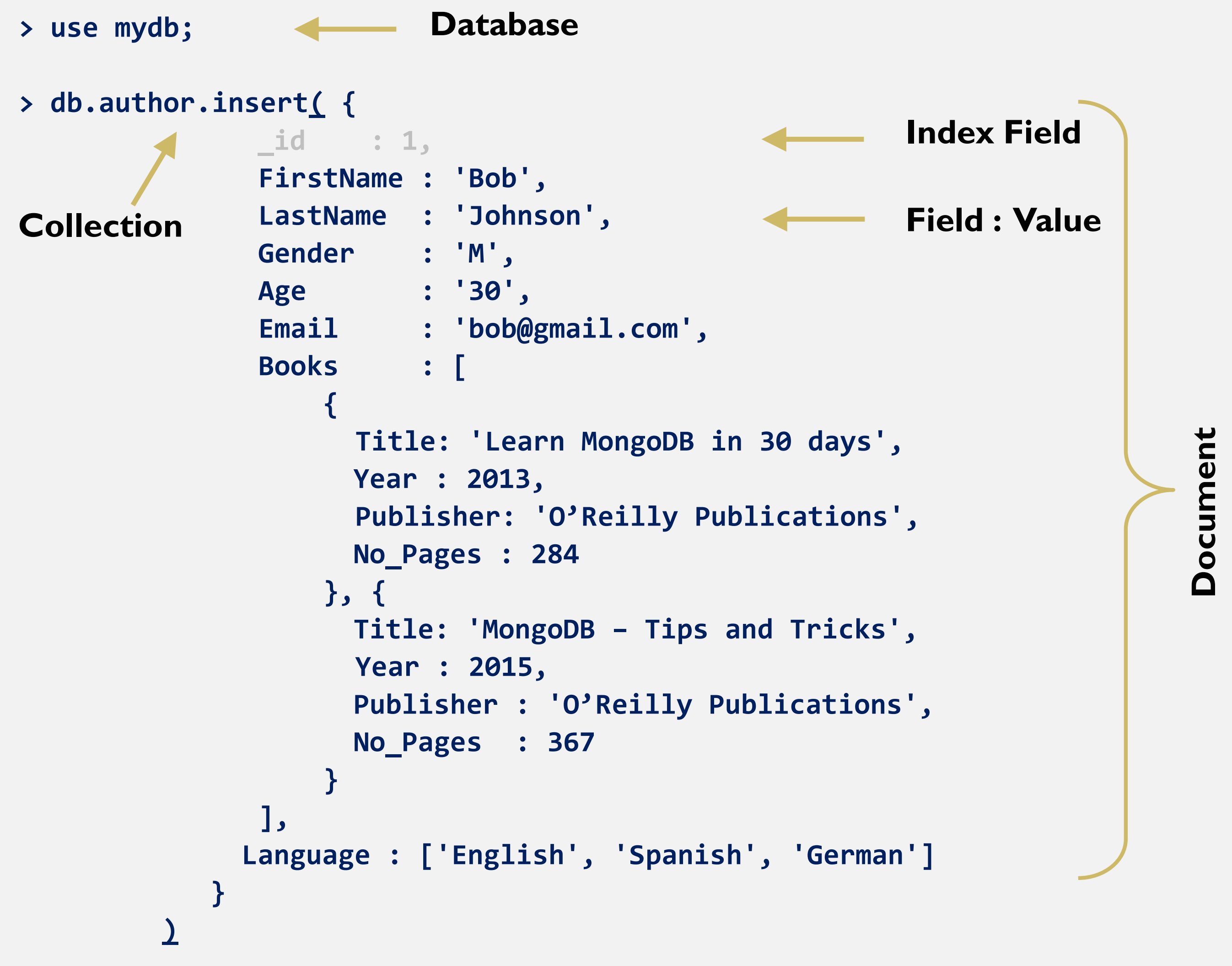


- Auto-Sharding
  - Range-based Sharding
  - Hash-based Sharding
  - Location-Aware Sharding
- Query Router
- Availability
- Consistency
  - ACID compliant and Document level

Query Language in Detail …

Query Language - Insert

```
Syntax:       db.collection.insert( {field:value, ... field:value} )
```



```
> use mydb;

> db.author.insert( {
               _id     : 1,
               FirstName : 'Bob',
               LastName  : 'Johnson',
               Gender    : 'M',
               Age       : '30',
               Email     : 'bob@gmail.com',
               Books     : [
                   {
                     Title: 'Learn MongoDB in 30 days',
                     Year : 2013,
                     Publisher: 'O’Reilly Publications',
                     No_Pages : 284
                   }, {
                     Title: 'MongoDB – Tips and Tricks',
                     Year : 2015,
                     Publisher : 'O’Reilly Publications',
                     No_Pages  : 367
                   }
               ],
              Language : ['English', 'Spanish', 'German']
            }
         )
```

Query Language - Update

`Syntax:      db.collection.update( {conditions}, {update_fields}, {option} )`

```
> db.author.update (
              {Email       : 'bob@gmail.com'},
              {FirstName   : 'Rob'},
              {upsert      : true}
        )    // replaces the complete document with new fields
```

```
> db.author.update (
              {Email       : 'bob@gmail.com'},
              {$set: {FirstName : 'Rob',
                      LastName  : 'Tom'}
              },
              {upsert      : true}
        )
```

```
> db.author.update (
              {Books.Title : 'MongoDB – Tips and Tricks'},
              {FirstName    : 'Rob'},
              {$set: {Year : 2016,
                      No_Pages  : 167}
              }
        )
```

Query Language - Delete

```
Syntax:       db.collection.deleteOne ( {conditions} )
              db.collection.deleteMany( {conditions} )
              db.collection.remove    ( {conditions} )
```

```
> db.author.deleteOne ( {Email : 'bob@gmail.com'} )
> db.author.remove( {Email : 'bob@gmail.com'}, 1 )
```

```
> db.author.deleteMany( {Email : 'bob@gmail.com'} )
> db.author.remove( {Email : 'bob@gmail.com'} )
```

```
// Removes all the documents from collection ‘author’
> db.author.deleteMany({ })
> db.author.remove({ })
```

Drop

```
Syntax:       db.collection.drop()
```

```
> db.author.drop( )
```

Query Language - Select

Syntax: db.collection.find ({condition}, {fields_to_return})

```
> db.author.find ()  //Returns all the fields and documents
> db.author.find( { Age: 30 } ) //Returns all the fields and documents where Age=30
> db.author.find( { Age: {$gt : 30} } )
> db.author.find( { Age: {$gt: 25, $lt: 55} } )
> db.author.find( { Books.Year: 2015, Books.No_Pages: 367 } )
```

```
//Returns _id, FirstName, LastName of all documents
> db.author.find ({}, {FirstName : 1, LastName : 1})


//Returns FirstName, LastName of documents where Age=30
> db.author.find( { Age: 30 }, {_id : 0, FirstName : 1, LastName : 1} )
```

Query Language - Index

Syntax: db.collection.createIndex({index_fields}, {options})

```
//Unique Index in ascending order
> db.author.createIndex( { Email: 1 }, { unique: true } )


//Compound Index in ascending order
> db.author.createIndex( { Email: 1, Lastname: 1, Firstname: 1 }, { unique: true } )


//Compound Index. Email in ascending order and Age in descending order
> db.author.createIndex( { Email: 1, Age: -1 })


//Unique Index in ascending order
> db.author.createIndex( { Books.Title: 1 } )
```

Query Language – TTL Index

```
Syntax:       db.collection.createIndex( {date_field}, {expireAfterSeconds: x})
```

```
//TTL Index. Document expires after 30 days from creation date (30 x 24 x 60 x 60)
> db.tempstore.createIndex({createDate : 1}, {expireAfterSeconds: 2592000})


//TTL Index. Document expires at future expiry date
> db.tempstore.createIndex({expiryDate : 1}, {expireAfterSeconds: 0})
```

Query Language – Geospatial Index

Syntax: db.collection.createIndex({location_field: "2dsphere"})

```
{      _id  : 1,
       name : "Apple Store",
       city : "Palo Alto",
       //geojson document. Longitude, Latitude. /Other types: Line, LineString,
       //Polygon, MultiPoint, MultiLineString, MultiPolygon, GeometryCollection
       location : {"type": "Point", "coordinates": [-122.1691291, 37.4434854] },
       type : "Retail"
}
{      _id  : 2,
       name : "Peninsula Creamery",
       city : "Palo Alto",
       location : {"type" : "Point",     "coordinates" : [-122.158428, 37.440675] },
       type : "Restaurant"
}
{      _id  : 3,
       name : "Fry's Electronics",
       city : "Palo Alto",
       location : {"type" : "Point",     "coordinates" : [-122.137044, 37.423556] },
       type : "Retail"
}
```

```
//Geospatial Index
> db.places.createIndex( {location : "2dsphere"} )
```

Query Language – Geospatial Search

```
// Below query outputs all documents within 2000M radius of given coordinates
db.places.find( {
   location: {
      $near: { $geometry: { type: "Point", coordinates: [-122.166641, 37.4278925] },
              $maxDistance: 2000  // distance in Meters to be searched
      }
   }
} )
```

MongoDB – Processes

```
// Starts Primary Daemon process (Server thread)
>  mongod

// Starts shard routing process
>  mongos

// Starts interactive Javascript Scripting Window
>  mongo
```

Hands-On

THANK YOU!