CS 572 Modern Web Applications

Najeeb Najeeb, PhD (najeeb@miu.edu)

V3.0.0

JavaScript Full Stack Development

- MongoDB
  - NoSQL database (document store)
  - Stores JSON documents
- Express
  - JavaScript web framework
  - On top of Node
- Angular
  - JavaScript UI framework
  - Single Page Applications
- Node
  - JavaScript server-side platform
  - Single threaded, fast and scalable

Roadmap and Outcomes

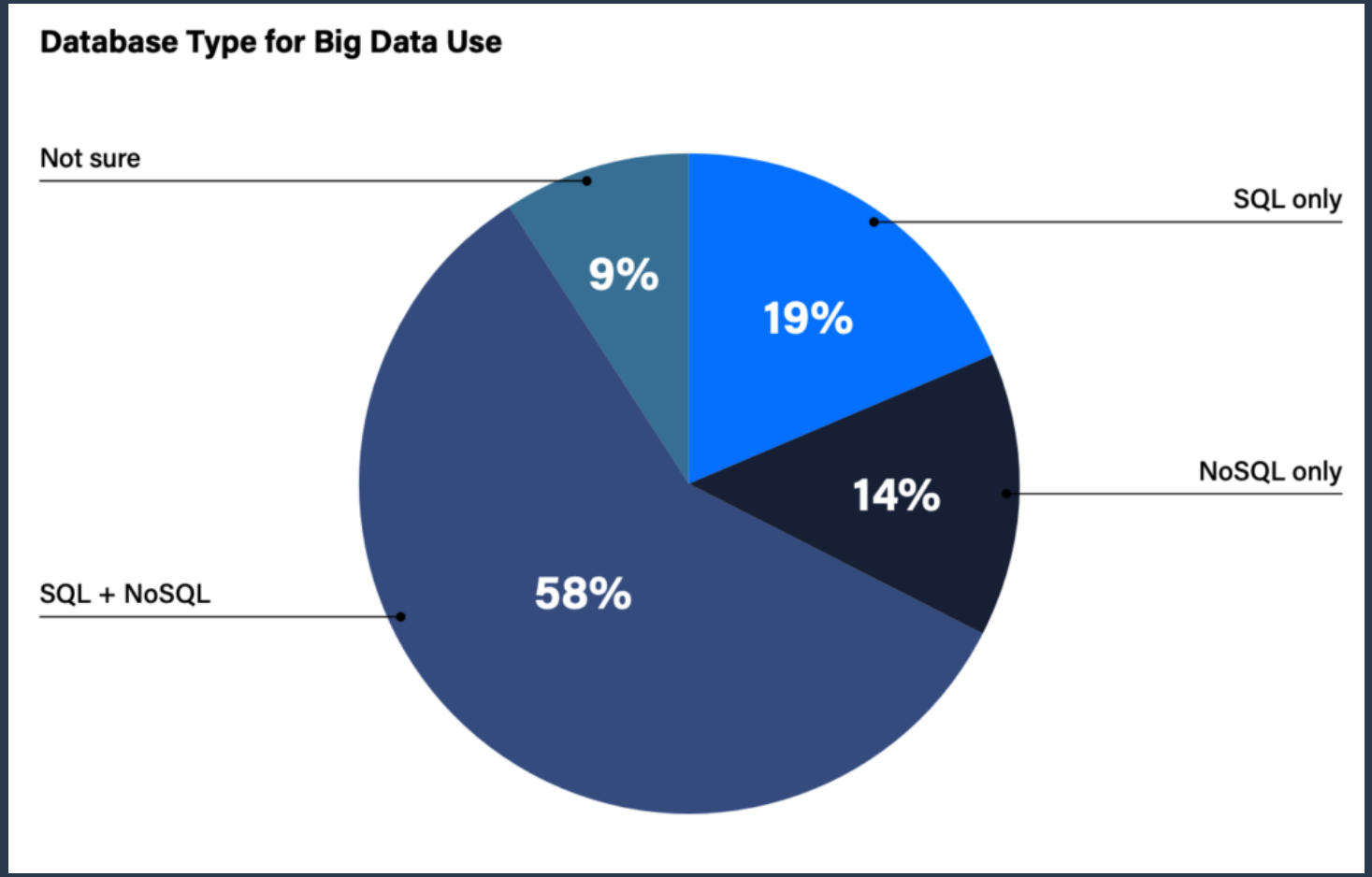
- Node.js: write asynchronous (non-blocking) code. Understand node platform to start a project.
- Express: setup express and get requests and send back responses. REST API.
- MongoDB: what NoSQL DB looks like. Full API interacting with DB.
- Angular: Investigate Angular and the architecture of an Angular application. Build a single-page application.
- MEAN application: Learn by example. We will create a MEAN Games application.

SQL vs NoSQL

Market Shares

NoSql vs. NewSQL vs Distributed SQL: DZone's 2020 Trend Report

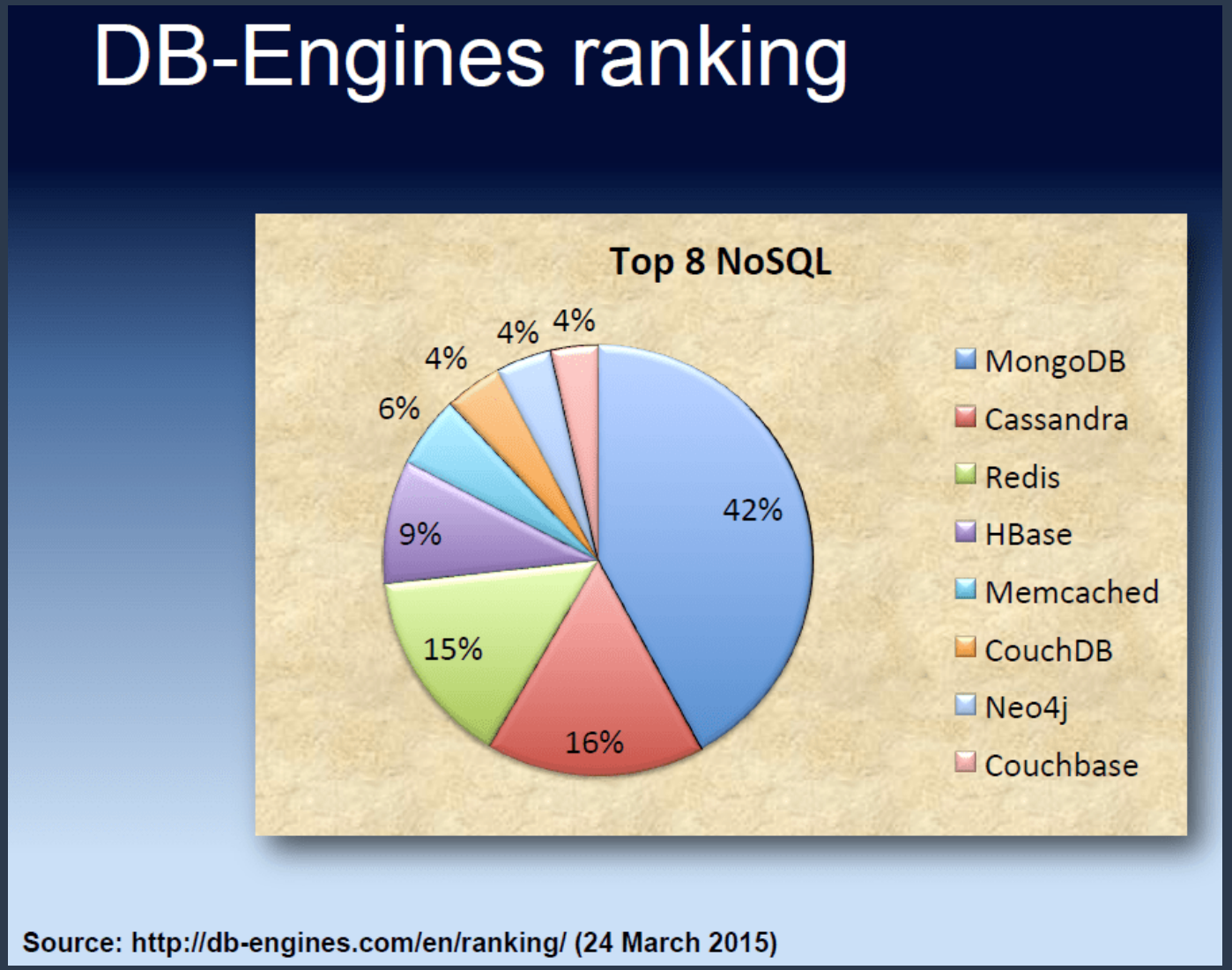
Written by Charlotte Dillon on Sep 9, 2020



https://www.cockroachlabs.com/blog/dzone-sql-trend/

NoSQL Databases

The NoSQL market size was valued at $2,410.5 million in 2018, and is projected to reach $22,087 million by 2026, growing at a CAGR of 31.4% from 2019 to 2026.

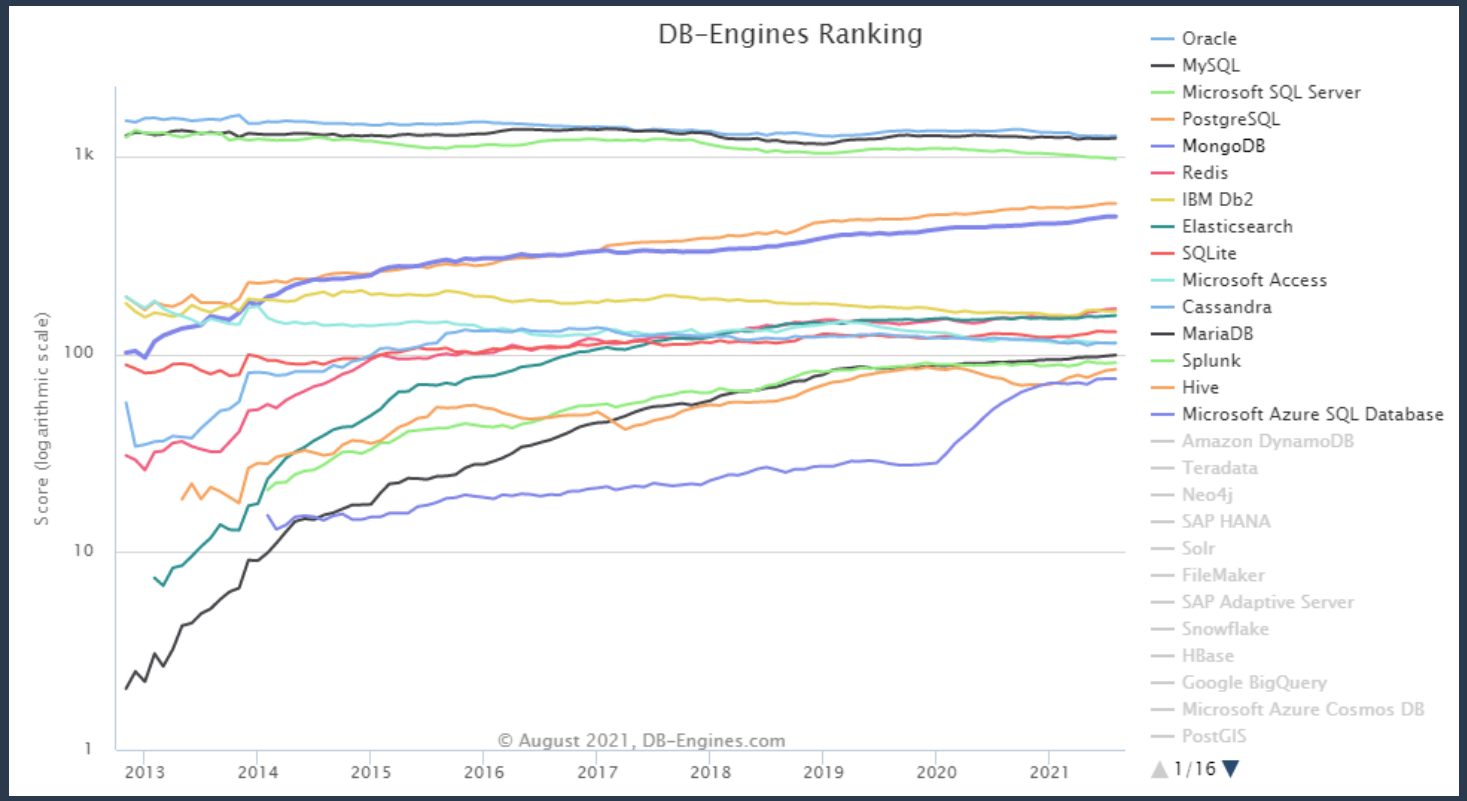


NoSQL Databases

DB-Engines Ranking – Trend Popularity.

August 2021

[historical trend of the popularity ranking of database management systems (db-engines.com)]



Introducing NoSQL-DB, MongoDB
Order is Present Everywhere

Wholeness

Data is the center of your application. The way data is presented impacts how it should be stored. Designing data storage to match its presentation creates more efficient applications. The Universe is structured in hierarchical layers from concrete expressions to their abstract basis, life is rich, and nature is efficient because of the underlying universal principles.

Introducing NoSQL-DB MongoDB
Order is Present Everywhere

1. What is MongoDB?
2. How to use MongoDB?
3. Best practices, and why?

Introducing NoSQL-DB MongoDB
Order is Present Everywhere

1. What is MongoDB?
2. How to use MongoDB?
3. Best practices, and why?

NoSQL DB

NoSQL Database Types

- Key-value store, ArangoDB
  - Store unique key and value, high scalability for caching (session management)
- Document store, MongoDB
  - Store semi-structured data in document format, no schema insert (mobile applications)
- Wide- column store, Amazon DynamoDB
  - Store in columns not rows, fast (catalogs, recommendation engines)
- Graph databases, Amazon Neptune
  - Store data as nodes and edges, show connections (reservation systems)
- More

Document Store vs Relational DB

RELATIONAL DB

STUDENT_ID	NAME	GPA
1	Jack	3.0
2	Jill	3.3
3	John	2.8

ID	COURSE_NAME	STUDENT_ID
1	Software Engineering	1
2	Web Programming	2
3	Algorithms	2

DOCUMENT STORE

```
[
 { "StudentID" : 1,
   "Name" : "Jack",
   "GPA" : 3.0,
   "Courses" : [
    { "ID" : 1,
      "CourseName" : "Software Engineering" } ] },
 { "StudentID" : 2,
   "Name" : "Jill",
   "GPA" : 3.3 ,
   "Courses" : [
    { "ID" : 2,
      "CourseName" : "Web Programming" },
    { "ID" : 3,
      "CourseName" : "Algorithms" } ] },
 { "StudentID" : 3,
   "Name" : "John",
   "GPA" : 2.8 }
]
```

NoSQL DB Design

- What is all the data you wish to output (at once) on a pages.
  - Put that information in one place.
- If on some page you wish to display some of the information from another document.
  - Add what needs to be displayed and include an ID to link to the other document.
- Optimize for the most common operation.
  - Reduce updates for the most common changeable items.
  - Increase speed of displaying most common pages.
- Keep number of Collections (Tables) to a minimum.
- Try to reduce each page to one collection (or minimum number of joined collections)
- Most common operations must run faster (even at the expense of less common operations)

MongoDB

MongoDB Collections

REVIEW.JSON


```
[
{ "ReviewID" : 1,
  "Title" : "Good Game.",
  "Review" : "I enjoyed the game.",
  "Stars" : 4,
  "Game" : {
    "ID" : 1,
    "Name" : "Trains"}
},
{ "ReviewID" : 2,
  "Title" : "Too Long.",
  "Review" : "The game is nice, but it was too long.",
  "Stars" : 3,
  "Games" : {
    "ID" : 2,
    "Name" : "Monopoly"}
}]
```


GAME.JSON


```
[{ "ID" : 1,
   "Name" : "Trains",
   "Price" : 48.82,
   "MinPlayers": 2,
   "MaxPlayers": 4,
   "EstimatedTimeToPlay": 45,
   "ReleaseYear": 2013},
{  "ID" : 2,
   "Name" : "Monopoly",
   "Price" : 29.97,
   "MinPlayers": 2,
   "MaxPlayers": 8,
   "EstimatedTimeToPlay": 180,
   "ReleaseYear": 1933},
{  "ID" : 3,
   "Name" : "Risk",
   "Price" : 20.99,
   "MinPlayers": 2,
   "MaxPlayers": 6,
   "EstimatedTimeToPlay": 120,
   "ReleaseYear": 1959}
]
```

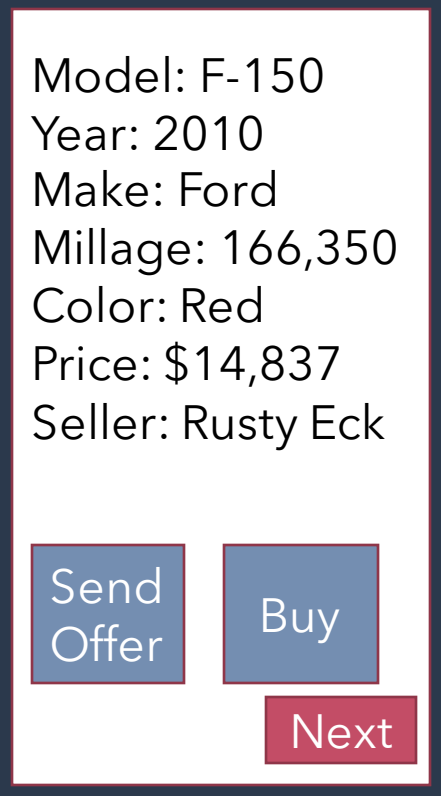
Learning Activity

- Break into groups (4 individuals in each group). Group ID will be distributed.
- Given the application X design a MongoDB database.
  - Identify the collection(s).
  - Identify the document layout in each collection.
- Then consider application Y, and state if it should use the same database or a different one and why.
- Even Group ID X= 1, Y=2
- Odd Group ID X=2, Y=1

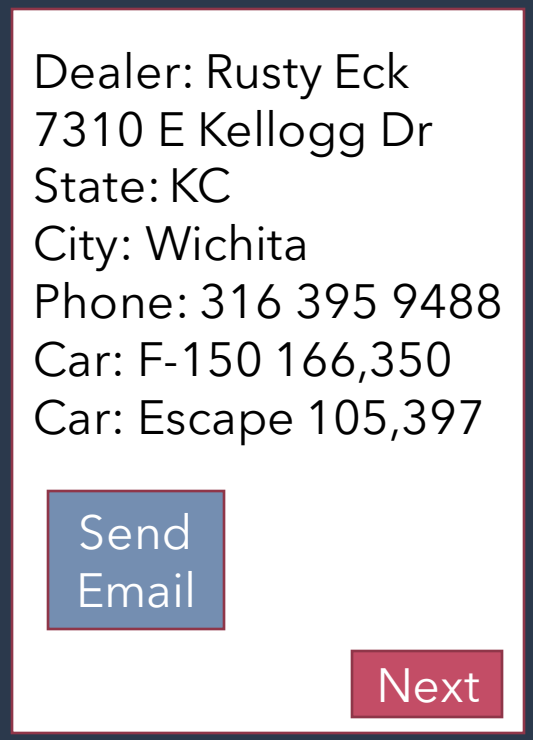
MongoDB Design

Design DB for application X, state if it can be used for application Y

APPLICATION 1

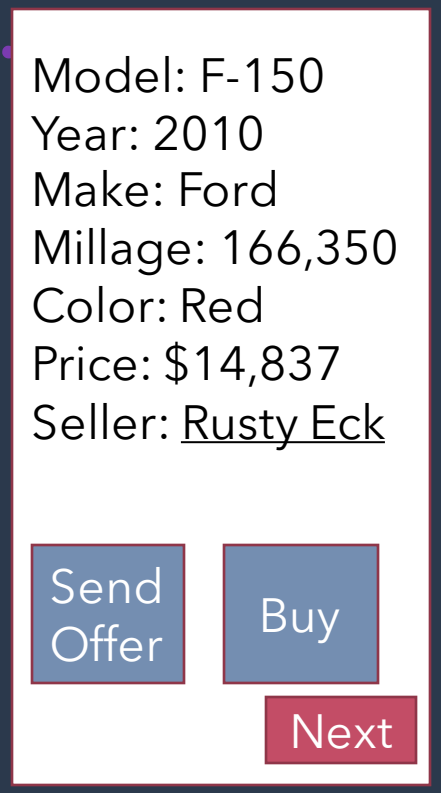


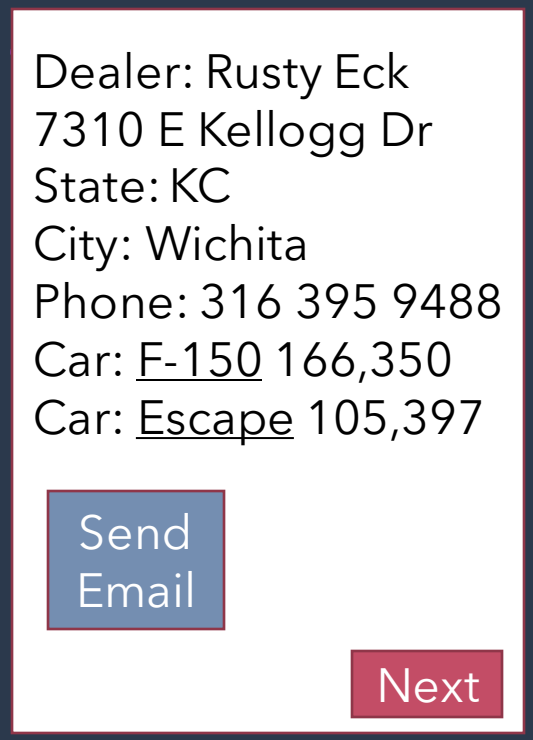
APPLICATION 2



MongoDB Design

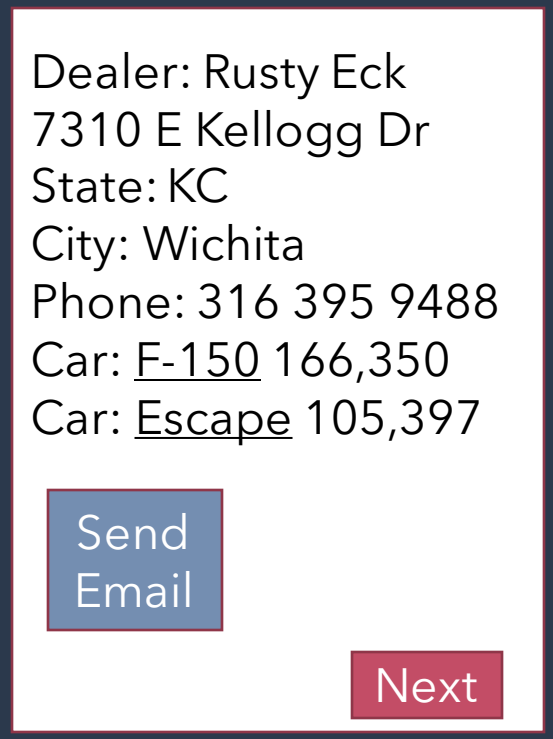
One Application Simple Decision?



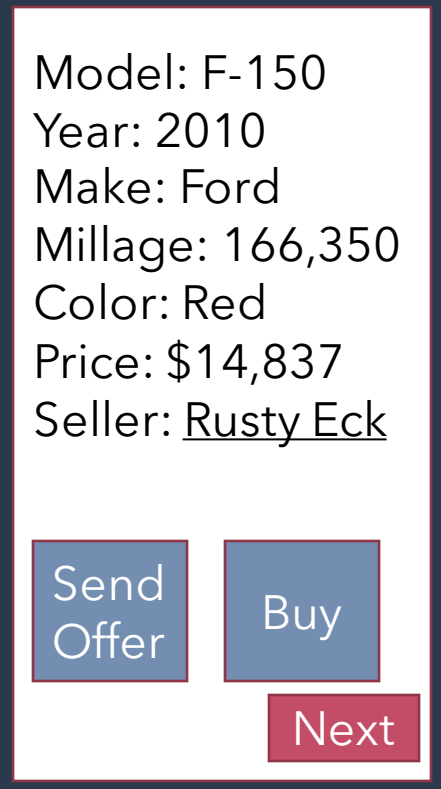


MongoDB Design

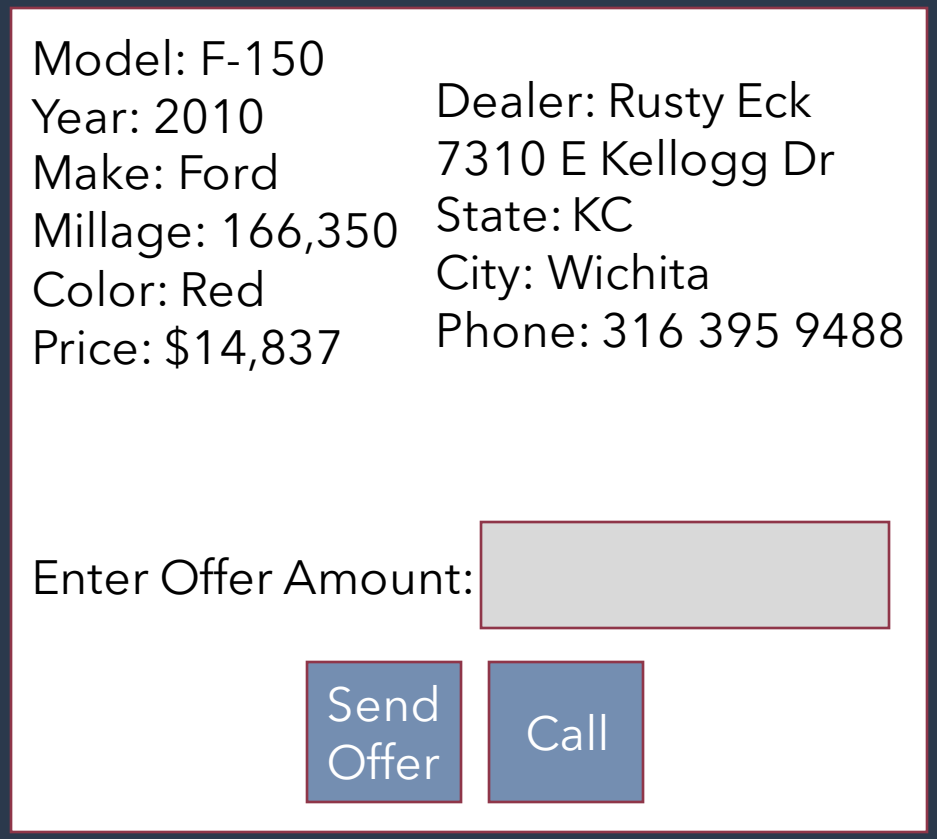
One Application Hard Decision?



Percentage of usage (Profile)
20%



Percentage of usage (Profile)
50% (20%)



Percentage of usage (Profile)
30% (60%)

How to Design a Document
Students' views

- Always try to design your collections as simple as possible.
- Design based on the UI (do not include information not visible in the UI)
- Reduce data redundancy (if data exists in another collections think of maybe using it? Depending on profiling).

MongoDB Collections

REVIEW.JSON


```
[
{ "ReviewID" : 1,
  "Title" : "Good Game.",
  "Review" : "I enjoyed the game.",
  "Stars" : 4,
  "Game" : {
    "ID" : 1,
    "Name" : "Trains"}
},
{ "ReviewID" : 2,
  "Title" : "Too Long.",
  "Review" : "The game is nice, but it was too long.",
  "Stars" : 3,
  "Games" : {
    "ID" : 2,
    "Name" : "Monopoly"}
}]
```


GAME.JSON


```
[{ "ID" : 1,
   "Name" : "Trains",
   "Price" : 48.82,
   "MinPlayers": 2,
   "MaxPlayers": 4,
   "EstimatedTimeToPlay": 45,
   "ReleaseYear": 2013},
{  "ID" : 2,
   "Name" : "Monopoly",
   "Price" : 29.97,
   "MinPlayers": 2,
   "MaxPlayers": 8,
   "EstimatedTimeToPlay": 180,
   "ReleaseYear": 1933},
{  "ID" : 3,
   "Name" : "Risk",
   "Price" : 20.99,
   "MinPlayers": 2,
   "MaxPlayers": 6,
   "EstimatedTimeToPlay": 120,
   "ReleaseYear": 1959}
]
```

How to Design a Document

- Why not have one Collection and store everything in it?
  - Not good logically and performance.
  - Hard to maintain.
- A review is for a game, so why not only have one Collection of Games.
  - A review can exists by itself.
  - Get all positive reviews, negative, …
  - A Game could also have several reviews.
- Collections may reference each other.
- You do not use a collection to get data from another collection.
  - What you want from another collection embed in your collection.

JSON and BSON

- JSON is what you use in your application.
- JSON is a close representation of what MongoDB stores.
- BSON is Binary-JSON, it is what MongoDB uses.
- BSON not human readable but maintains the flexibility and ease of use of JSON plus the speed of binary format.
- MongoDB accepts JSON and returns JSON (but stores it as BSON).

JSON ID

- MongoDB creates unique ID for a document when created.
- _id property is what MongoDB creates.
- The value is ObjectId("5f9aef68980db44d37c1aaed") unique combination of time (Unix epoch) , machine ID, process ID, and counter.

Main Points
Introducing NoSQL-DBs, MongoDB
Order is Present Everywhere

1. MongoDB is a document-based NoSQL database. It is a schema-less database, but each diverse and similar document carries its schema. Science and Technology of Consciousness: All diverse aspects of nature get unified at the Unified Field level. Everyone can experience this Unified Field.

Introducing NoSQL-DB MongoDB
Order is Present Everywhere

1. What is MongoDB?
2. How to use MongoDB?
3. Best practices, and why?

Install and Work With MongoDB

- Install from MongoDB website (www.mongodb.com/try/download)
- Running MongoDB
  - mongo --version
  - mongo
    - Download and install mongosh (https://www.mongodb.com/try/download/shell)
  - exit (or Ctrl + C)
- Create Database
- Create Collection
- Retrieve Collection

MongoDB
Database
Collection

List all databases on your system

```
show dbs
admin 0.000GB
config 0.000GB
local   0.000GB
```

Select database to work with

```
use local
switched to db local
```

Create new database, make sure it does not exist

```
use newTestDB
switch to newTestDB
```

Note: new database not created until you add a collection to it.

Get the current database being used

```
db (or db.getName();)
newTestDB
```

Delete database

```
db.dropDatabase();
{ "dropped" : "newTestDb", "ok" : 1}
```

MongoDB
Database
Collection

List collections in current database

```
use local
show collections
startup_log
use newTestDB
show collections
```

Create collection

```
db.createCollection("technology")
{ "ok" : 1}
```

Delete collection

```
db.technology.drop()
true
```

CRUD

Create

Read

Update

Delete

Add document in current collection

```
db.technology.insertOne(
... {
... name : "MongoDB",
... role : "Database"
... }
... );
{acknowledged: true, insertedId: ObjectId...}
```

List documents in current collection

```
db.technology.find();
{ "_id" : ObjectId("5f9aef68980db44d37c1aaed"), "name" :
"MongoDB", "role" : "Database" }
```

Insert multiple documents at once

```
db.technology.insertMany([{name : "Express", role: "Web
application server"},
... {name : "AngularJS", role: "Front-end framework"},
... {name : "Node.js", role: "Platform"}]);
{acknowledged: true, insertedIds: {...}}
```

CRUD

Create

Read

Update

Delete

List all documents in current collection

```
db.technology.find();
{ "_id" : ObjectId("5f9aef68980db44d37c1aaed"), "name" : "MongoDB", "role" : "Database" }
```

List based on document id in current collection

```
db.technology.find({"_id" :  ObjectId("5f9aef68980db44d37c1aaed")});
{ "_id" : ObjectId("5f9aef68980db44d37c1aaed"), "name" : "MongoDB", "role" : "Database" }
```

List based on name in current collection

```
db.technology.find({"name" :  AngularJS")});
{ "_id" : ObjectId("5f9af651980db44d37c1aaef"), "name" : "Angular", "role" : "Front-end framework" }
```

Sorting, 1 for assending –1 for decending

```
db.technology.find().sort({"name"  : 1});
```

Limit returned fields, projection (the second parameter in find).

```
db.technology.find({}, {"name" : true});
db.technology.find({}, {"name" : true, "_id" : false});
```

CRUD

Create

Read

Update

Delete

Update a document , finds the documents of interest the updates them. The first parameter is the query, the second is the data to set.

```
db.technology.updateOne( {"name" : "AngularJS"}, {$set : {"name" : Angular"} } );
{acknowledged: true,...,modifiedCount: 1,...}
```

Update more than one document at once

```
db.technology.updateMany({},{$set:{"language":JavaScript"}},{multi:true} );
{ acknowledged: true,
  insertedId: null,
  matchedCount: 4,
  modifiedCount: 4,
  upsertedCount: 0,
}
```

CRUD

Create

Read

Update

Delete

Delete document from collection, you provide a query object

```
db.technology.deleteOne( { "name" : "Express" })
{ acknowledged: true, deletedCount: 1 }
```

```
db.technology.deleteMany( {})
{ acknowledged: true, deletedCount: 3 }
```

This will remove all the documents from the collection :(

Import & Export Data

BSON

Import

Export

Import MongoDB data from BSON file

```
mongorestore --gzip dump\
...
31 document(s) restored successfully. 0 document(s)
failed to restore.
```

```
mongorestore --nsInclude=meanGames.games --
gzip dump\
...
29 document(s) restored successfully. 0
document(s) failed to restore.
```

BSON

Import

Export

Export MongoDB data as BSON file

```
mongodump --db meanGames
...writing meanGames.users to dump\meanGames\users.bson
...done dumping meanGames.users (2 documents)
...writing meanGames.games to dump\meanGames\games.bson
...done dumping meanGames.games (29 documents) dumping
newTestDb.technology (4 documents)
```

Compress the BSON output data

```
mongodump --db meanGames --gzip

...writing meanGames.users to dump\meanGames\users.bson.gz
...done dumping meanGames.users (2 documents)
...writing meanGames.games to dump\meanGames\games.bson.gz
...done dumping meanGames.games (29 documents)
```

JSON

Export

Import

Export MongoDB data as JSON file (for a collection only)

```
mongoexport --db meanGames --collection users
...connected to: mongodb://localhost/
{"_id":{"$oid":"5fd953feafb225d78c313b89"},"username":"jack2020
","name":"Jack","password":"$2a$10$tk5yi88QFlCerfVpDegmK.QHt
1suzL9p8XAQE2mNCVmPOHjzddqwG","__v":0}
{"_id":{"$oid":"5fd99007f33650e6f0a9999b"},"username":"Jim2020"
,"name":null,"password":"$2a$10$do.Uj1B/vu5ucyu8EQookeerw.Sk
R6/DdMoOfxAwQr3pJpvOFpeA.","__v":0}
...exported 2 records
```

Export to file

```
mongoexport --db meanGames –-collection users --out
output/game-users.json
exported 2 records
```

Export as an array

```
mongoexport --db meanGames –-collection users –out
output/game-users.json --jsonArray --pretty
exported 4 records
```

JSON

Export

Import

Import MongoDB data from JSON file

```
mongoimport --db meanGames --collection users --
jsonArray output/technology.json
imported 2 documents
```

Connecting MongoDB to NodeJS

MongoDB & NodeJS

- Installing mongoDB driver in our app.
- Creating reusable connections.
- Defining connection string.
- Accessing connections from controllers.
- Best practices while doing all this.

Connect to DB

Install driver

Connections

Use DB

Install MongoDB native driver

```
npm install mongodb --save
```

```
mongodb@4.2.2 node_modules/mongodb
```

Connect to DB

Install driver

Connections

Use DB

Create file to manage connections,
File api/data/dbconnection.js

```
const MongoClient= requires("mongodb").MongoClient;
let _connection= null;
const open= function() {
    if (get() == null)
        MongoClient.connect(process.env.DB_URL, function(err, client){
            if(err) {
                console.log("DB connection failed");
                return;
            }
            _connection= client.db(process.env.DB_NAME);
            console.log("DB connection open", _connection);
        });
    }
}
const get= function() {
    return _connection;
}
module.exports= {
    open : open,
    get : get
};
```

Update .env file

```
DB_URL= "mongodb://localhost:27017/meanGames"
DB_NAME= "meanGames"
```

Connect to DB

Install driver

Connections

Use DB

Open the connection as soon as the application starts, app.js

```
require("./api/data/dbconnection.js).open();
```

Run

```
npm start
```

DB connection open ....

Check for error, change the port number in dbconnection.js and run again.

Connect to DB

Install driver

Connections

Use DB

Use the db connection in the controllers.
api/controllers/games.controllers.js

```
const dbConnection= require("../data/dbconnection");

... getAll= ..
const db= dbConnection.get();
console.log("db", db);
```

Run on browser (http://localhost:3000/api/games)

```
npm start
db ...
```

Opening db is asynchronous. So make sure you get it when you need it. Don't just open it at the start of the file.

Opening db connection is slow. Best to open it once at application start and reuse it.

No need for a global variable for db. Encapsulated in dbconnection.

Main Points
Introducing NoSQL-DBs, MongoDB
Order is Present Everywhere

1. MongoDB is a document-based NoSQL database. It is a schema-less database, but each diverse and similar document carries its schema. Science and Technology of Consciousness: All diverse aspects of nature get unified at the Unified Field level. Everyone can experience this Unified Field.
2. We use MongoDB driver to connect to a MongoDB instance. We must remember to create the DB connection only once (asynchronously) and then use it several times. Also, the connection must be available when needed. These steps may sometimes become tricky and complex. Science and Technology of Consciousness: In contrast, connecting to the Unified Field is effortless, simple, and easy. You only need a mantra and 20 minutes to connect to the source of knowledge.

Introducing NoSQL-DB MongoDB
Order is Present Everywhere

1. What is MongoDB?
2. How to use MongoDB?
3. Best practices, and why?

Working with MongoDB in NodeJS

Query DB

GetAll

Pagination

GetOne

Use the db connection in the controllers.

api/controllers/games.controllers.js

```
… getAll= ..
const gamesCollection= db.collection("games");
// const docs= gamesCollection.find(); //Sync not good :(
gamesCollection.find().toArray(function(err, docs) {
    console.log("Found games", docs);
    res.status(200).json(docs);
}
```

Run on browser (http://localhost:3000/api/games)

```
npm start
```

Found games …

Query DB

GetAll

Pagination

GetOne

Use the db connection in the controllers.

api/controllers/games.controllers.js

```
… getAll= ..
const gameCollection= db.collection("games");
let offset= 0;
letcount= 5;
if (req.query && req.query.offset) {
   offset= parseInt(req.query.offset, 10);
}
if (req.query && req.query.count) {
   count= parseInt(req.query.count, 10);
}
collection.find().skip(offset).limit(count).toArray(function(err, games) {
   console.log("Found games", games);
   res.status(200).json(games);
}
```

Run on browser (http://localhost:3000/api/games)

```
npm start
Found games …
```

Query DB

GetAll

Pagination

GetOne

Use the db connection in the controllers.
api/controllers/games.controllers.js

```
…
const ObjectId= require("mongodb").ObjectId;
…
getOne= function(req, res) {
…
    const db= dbConnection.get();
    const gamesCollection= db.collection("games");
    const gameId= req.params.gameId;
    gamesCollection.findOne({_id : ObjectId(gameId)}, function(err,
game) {
        console.log("Found game", game);
        res.status(200).json(game);
    }
…
```

Run on browser (http://localhost:3000/api/games)

```
npm start
Found games …
```

Insert DB

Error Checking

InsertOne

Use the db connection in the controllers.

api/controllers/games.controllers.js

```
const ObjectId= require("mongodb").ObjectId;
...
addOne= function(req, res) {
...
const db= dbConnection.get();
const gamesCollection= db.collection("games");
if (req.body && req.body.title && req.body.price) {
    console.log(req.body);
    res.status(200).json(req.body);
} else {
    console.log("Data missing from POST body");
    res.status(400).json({error : "Required data missing from POST"});
}
```

Run app.boomerangapi.com/workspace on the browser

```
npm start
```

error: "Required data missing from POST" …

Insert DB

Error Checking

InsertOne

Use the db connection in the controllers.
api/controllers/games.controllers.js

```
...
addOne= function(req, res) {
...
    let newGame= {};
    if (req.body && req.body.title&& req.body.price) {
        newGame.title= req.body.title;
        newGame.price= parseFloat(req.body.price);
        gamesCollection.insertOne(newGame, function(err, response) {
        if (err) {
            res.status(500).json({error: err});
        } else {
            console.log(response);
            res.status(201).json(response);
        }
    }
...
```

Run app.boomerangapi.com/workspace on the browser
npm start
Found games ...

MongoDB & NodeJS

- We will not be using mongoDB directly from nodeJS.
- There is a much easier way to work with mongoDB from Node.
- We will use Mongoose.

Main Points
Introducing NoSQL-DBs, MongoDB
Order is Present Everywhere

1. MongoDB is a document-based NoSQL database. It is a schema-less database, but each diverse and similar document carries its schema. Science and Technology of Consciousness: All diverse aspects of nature get unified at the Unified Field level. Everyone can experience the Unified Field.

2. We use MongoDB driver to connect to a MongoDB instance. We must remember to create the DB connection only once (asynchronously) and then use it several times. Also, the connection must be available when needed. These steps may sometimes become tricky and complex. Science and Technology of Consciousness: In contrast, connecting to the Unified Field is effortless, simple, and easy. You only need a mantra and 20 minutes to connect to the source of knowledge.

3. One best practice when working with MongoDB is to separate the creation of a connection (in a db.js file) from the DB operations (in the controllers). DB operations are asynchronous. So, we deal with results in callbacks. All this results in more efficient performance in database-driven programs. Science and Technology of Consciousness: Neuroscience, the scientific study of the human nervous system, verifies that efficient performance in activities is more dependent upon the coherent functioning of the brain than it does on education, work experience, and age.[1] TM creates coherent brain functioning.[2]

[1]: Travis, Frederick; and Arenander, Alarik. Cross-sectional and longitudinal study of effects of Transcendental Meditation practice on interhemispheric frontal asymmmetry and fontal coherence. International Journal of Neuroscience 116: 1519–1538, 2006.
[2]: Travis, Frederick; Grosswald, Sarina; and Stixrud, William. ADHD, brain functioning, and Transcendental Meditation practice. Mind & Brain, The Journal of Psychiatry 2: 73–81, 2011.