# Your Next Week

## Saturday March 21

*9 AM*
**— DUE Class 02 Code Challenge**
**— DUE Class 02 Lab**
**— DUE Class 03 Reading**
— Class 03

*MIDNIGHT*
**— DUE Class 03 Learning Journal**

## Sunday March 22

*MIDNIGHT*
**— DUE Career: Professional Etiquette**
**— DUE Class 02 - 03 Feedback**

## Monday March 23

## Tuesday March 24

*6:30 PM*
**— DUE Class 03 Code Challenge**
**— DUE Class 03 Lab**
**— DUE Class 04 Reading**
— Class 04A

## Wednesday March 25

*6:30 PM*
— Class 04B

*MIDNIGHT*
**— DUE Class 04 Learning Journal**

## Thursday March 26

*6:30 PM*
— Lab 04 & Code Challenge 04 Co-Working

## Friday March 27

## Saturday March 28

*9 AM*
**— DUE ALL PRE-WORK**
**— DUE Class 04 Code Challenge**
**— DUE Class 04 Lab**
**— DUE Class 05 Reading**
— Class 05

*MIDNIGHT*
**— DUE Class 04 Learning Journal**

# Class 03

# Data Modeling & NoSQL Databases

seattle-javascript-401n16

# Lab 02 Review

# Code Challenge 02 Review

# Databases

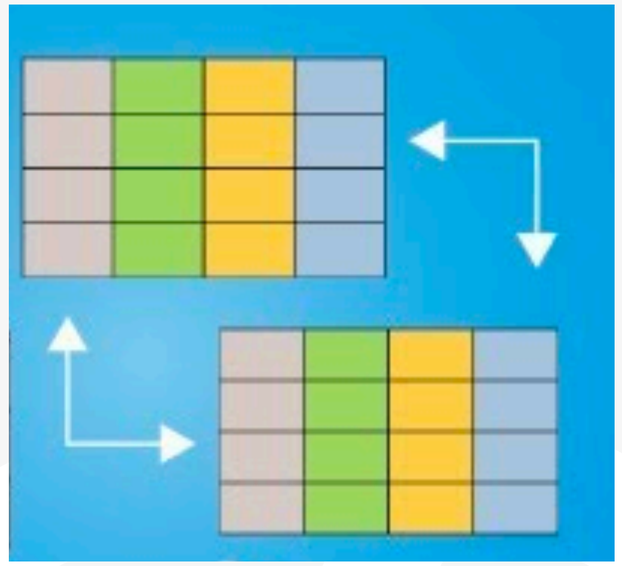
- So far, your notes application lets you:
    - “Add” a note (log it to the console)
    - Validate your command line arguments
    - Validate your created note
- Now we want to save these notes!
- Our application will send information to a database to store long term

# Aw CRUD!

- When we connect to a database, there are **four major actions** we want to do
  - **Create** an entry
  - **Read** a stored entry
  - **Update** a stored entry
  - **Delete** a stored entry
- We refer to these basic actions as “**CRUD**”

# SQL vs NoSQL

- Postgres is a **relational** database using **structured query language (SQL)**
  - Schema leads to creation of tables
  - If two things are related, another table needs to relate them
- We’re moving to MongoDB, a **non-relational**, **non-SQL (NoSQL)** database
  - Very fluid / flexible
  - Essential JSON data, connections are defined by us



# Our Data

```
const noteSchema = {
    id: { type: 'string', required: true },
    note: { type: 'string', required: true },
    categoryId: { type: 'string', required: false },
};
```

```
let myNote = {
    id: '001',
    note: 'Hello World!',
    categoryId: ['001'],
};
```

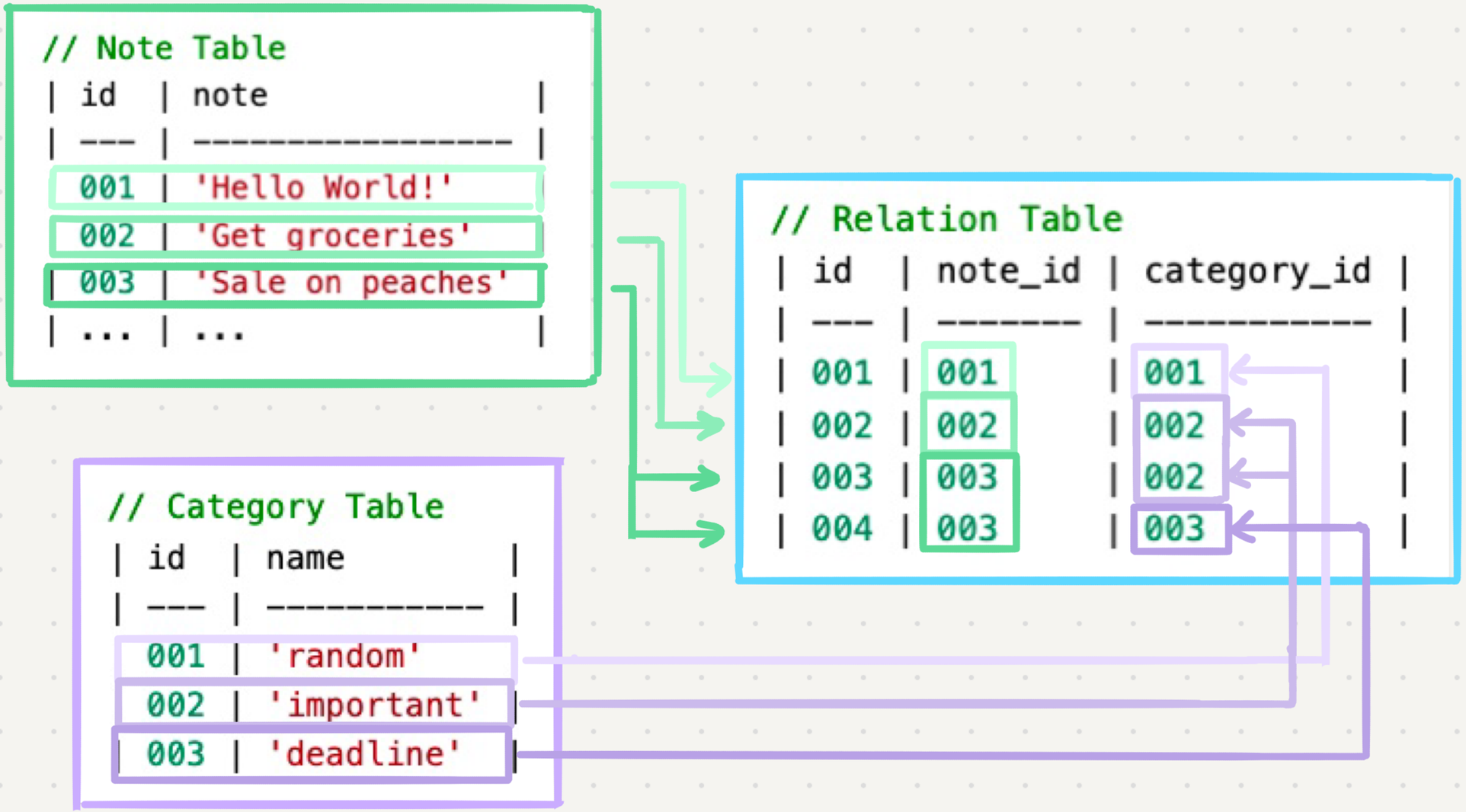
RELATION

```
let categoryRandom = {
    id: '001',
    name: 'random',
};
```

```
const categorySchema = {
    id: { type: 'string', required: true },
    name: { type: 'string', required: true },
};
```

```
let categoryImportant = {
    id: '002',
    name: 'important',
};
```

# Represented in SQL



# Represented in NoSQL

```
// Notes
[
    {
        id: '001',
        note: 'Hello World!',
        categoryId: ['001'],
    },
    {
        id: '002',
        note: 'Get groceries',
        categoryId: ['002'],
    },
    {
        id: '003',
        note: 'Sale on peaches',
        categoryId: ['002', '003'],
    },
];
```

NO ENFORCED RELATION

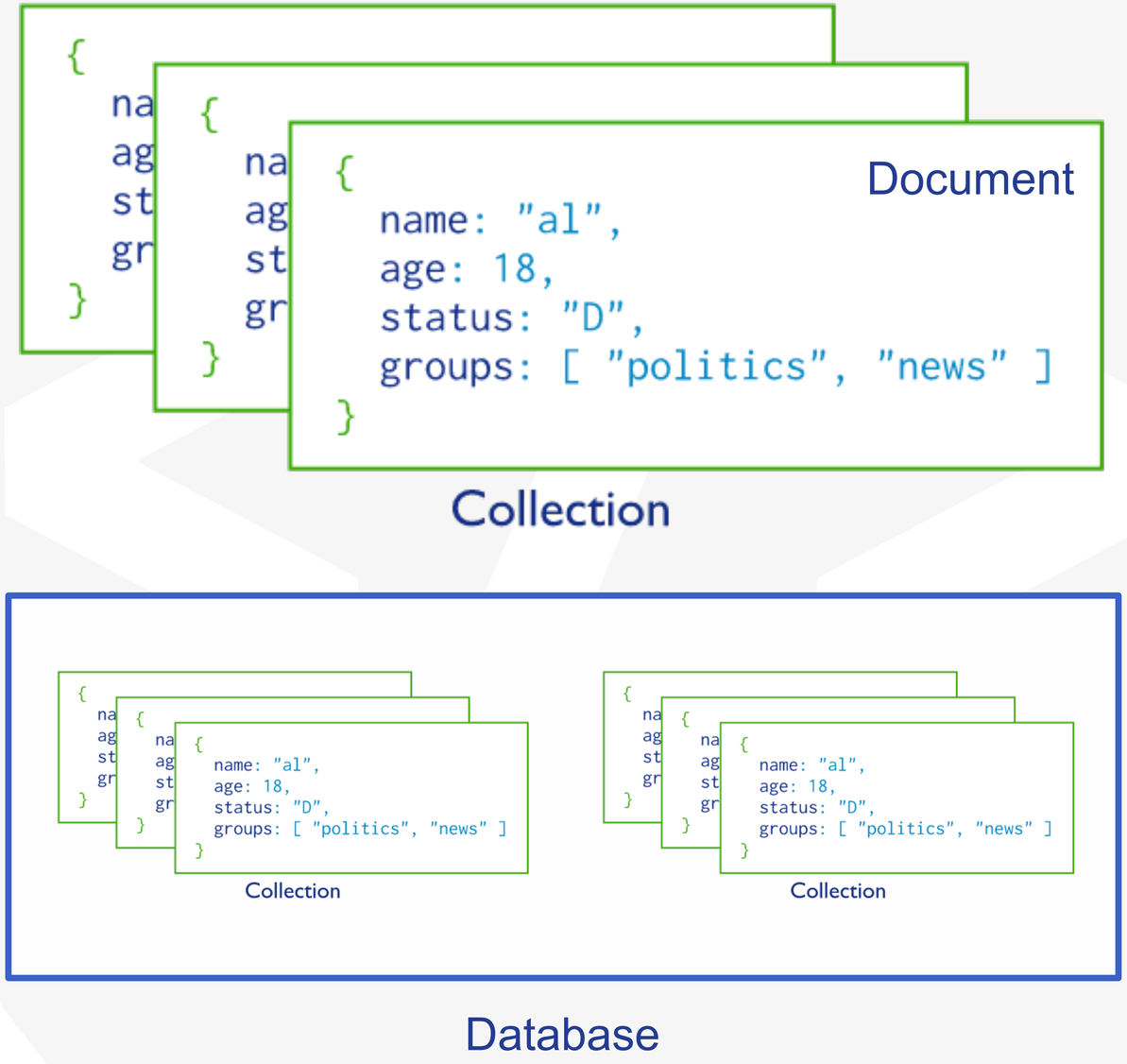
```
// Categories
[
    {
        id: '001',
        name: 'random',
    },
    {
        id: '002',
        name: 'important',
    },
    {
        id: '003',
        name: 'deadline',
    },
];
```

# MongoDB

- A very popular NoSQL database system
    - Flexible, scaleable, easy to setup
- Works really well with JavaScript
- Dependency: `npm install mongodb`
- API with many operations
    - More than just CRUD
- Can make both local and “production” databases

# Collections and Documents

- MongoDB has new terminology:
  - **Record** - An object representing a piece of data (aka document)
  - **Document** - An individual record for a given collection, stored as a `.bson` (binary JSON) file (aka record)
  - **Collection** - All the records for a specific schema (notes, categories)
  - **Database** - What contains all your data / all your collections



# Using MongoDB from CLI

## Running Local MongoDB

```
mongod --dbpath=[/PATH/TO/DATA/FOLDER]
```

## MongoDB Shell Commands

| Command | Description |
| --- | --- |
| `mongo` | Launch the mongo shell. Once in the shell, you should see `>` |
| `show dbs` | Show all the databases |
| `use db <name>` | Use the database with name `<name>` |
| `show collections` | Show all the collections in the current database |
| `db.<collection>.find()` | List all the documents / records in the specified collection `<collection>` |
| `db.<collection>.save()` | Save a new document / record to the specified collection `<collection>` |
| `db.<collection>.drop()` | Completely removes the specified collection `<collection>` |

# Other Ways to Use MongoDB

- MongoDB Atlas - Lets you create a remote database instead of a local one (useful for Windows users)
- mLab - What you'll use to create a "production" database when deploying to Heroku
- MongoDB Compass - A handy graphical user interface (GUI) for accessing database collections / documents