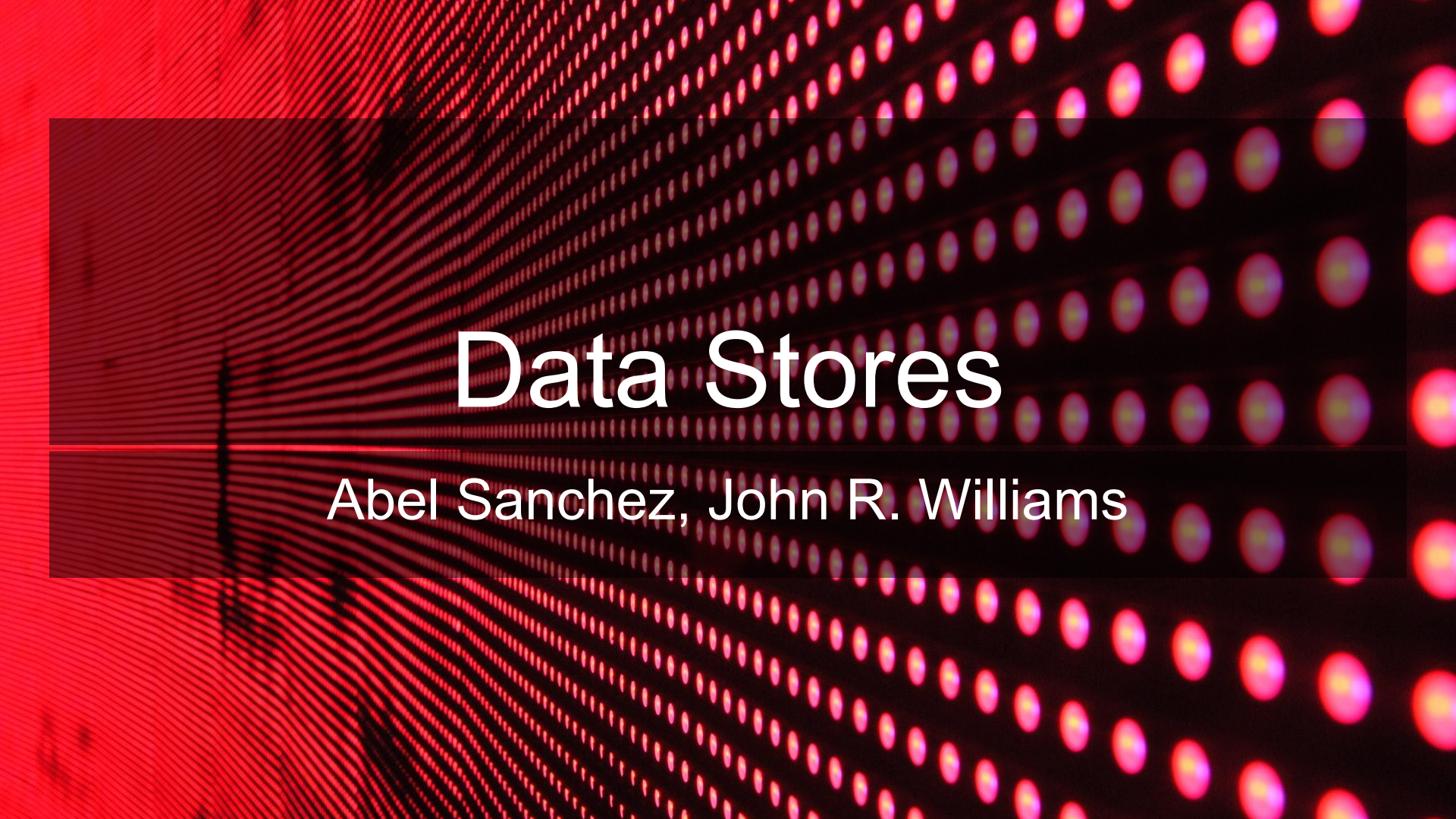


Data Stores

Abel Sanchez, John R. Williams

Data

Fundamental components of computer programs:

- A way to represent data
- A way to store data
- Instructions to manipulate data

Data Store

Stores and manipulates information

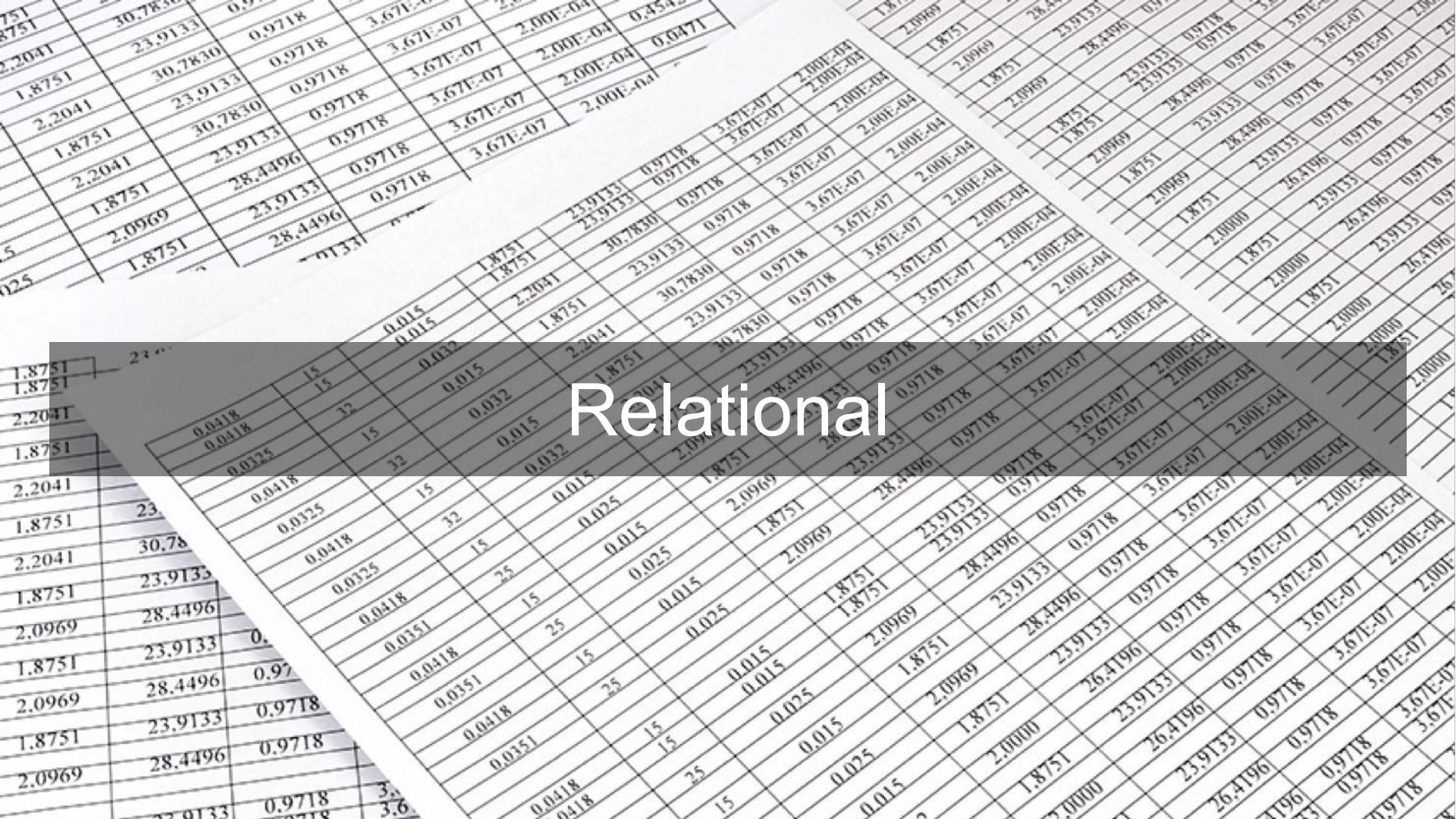
Two Broad Types of Data Stores

- Relational: the classic - the old - represents all data as tables
- NoSQL: alternatives to traditional database
  - We will see examples of document databases
    - Data is defined in terms of documents - JSON objects

Database Design

Database design involves choosing:

- The documents collections (tables)
- The fields/properties (the columns)
- How collections and properties interact


Relational

Entity-Relationship Modeling

E/R Modeling is used for conceptual design

- Entities - object or things of interest
- Attributes - facts of properties of an entity
- Relationships - links between entities

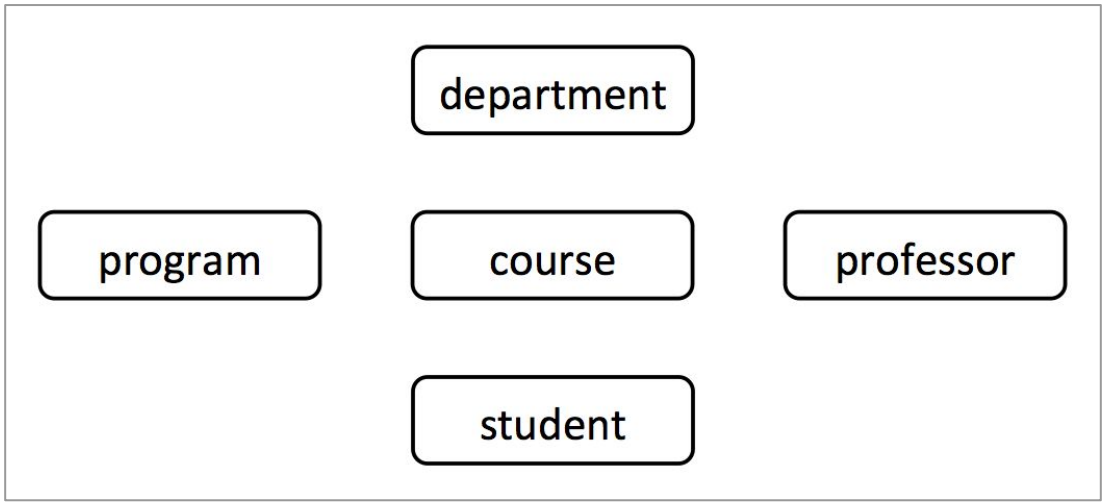
Entities

Represent objects or things of interest

- Physical things like students, professors, employees
- Abstractions like courses, programs, projects, orders
- Have types like course, professor
- Have instances like courses: calculus, algebra

Entities in Diagrams

Entities are drawn as a box



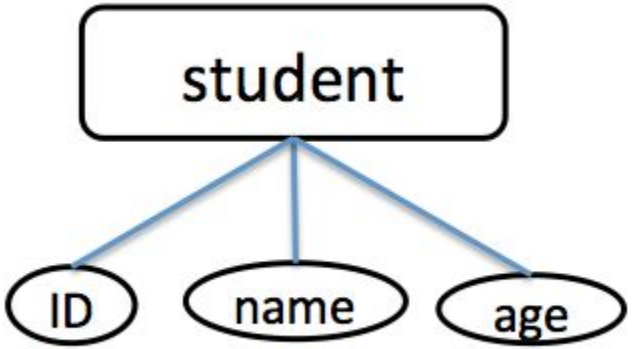
Attributes

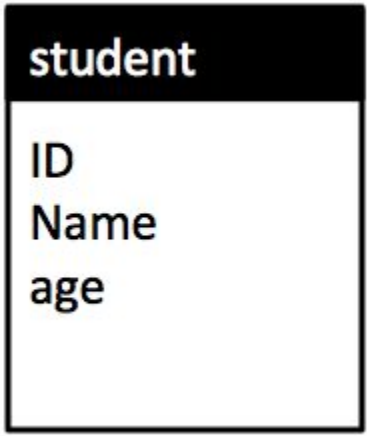
Attributes are facts, properties, or details of an entity.

- Students have IDs, names, courses
- Courses have names, credits, level

Attributes in Diagrams

Represented a few different ways





Relationships

Relationships are an association between two or more entities.

- A student takes several courses
- A course is taught by one professor

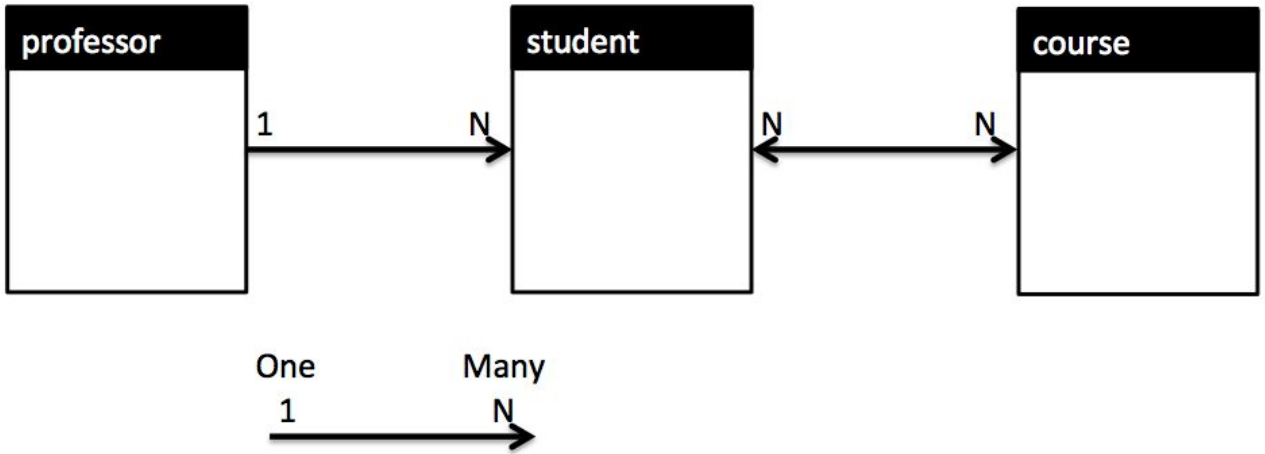
Cardinality Ratios

An entity can participate in three types of relationships.

- One to one (1:1). Each professor has one office
- One to many (1:N). A professor can advice many students
- Many to many (N:N). Each student takes many classes, and each class is taken my many students.

Relationships & Cardinality Ratios

Relationships are links between two entities



Entity-Relationship Modeling

Look at your data and identify

- Entities
- Attributes
- Relationships
- Cardinality ratios

Examples - University

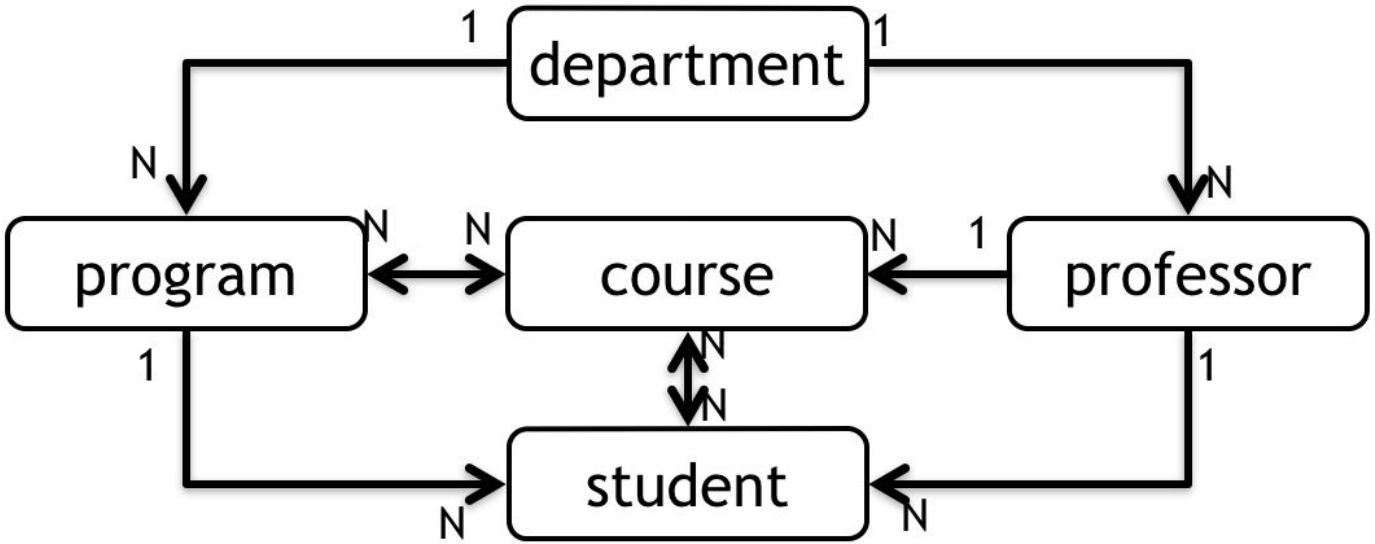
A university consists of a number of departments. Each department offers several programs. A number of courses make up each program. Students enroll in a particular program and take courses towards the completion of that program. Each course is taught by a professor from the appropriate department, and each professor advices a group of students.

Example - University entities

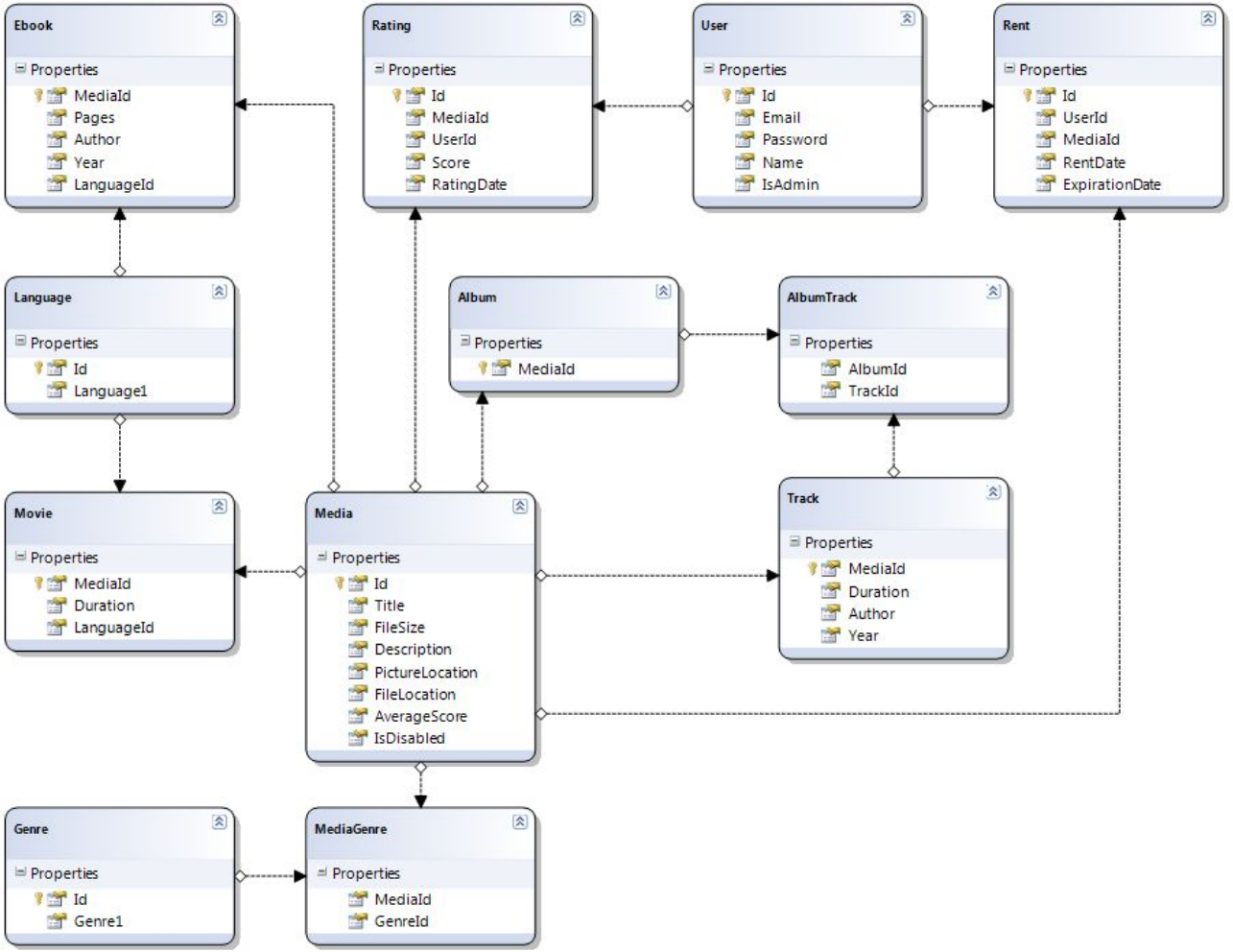
A university consists of a number of **departments**. Each department offers several **programs**. A number of courses make up each program. **Students** enroll in a particular program and take courses towards the completion of that program. Each course is taught by a **professor** from the appropriate department, and each professor advices a group of students.

Example - ER Diagram

Entities: department, program, course, professor, student



Sample: Media ER



Document Data Stores

Document data stores

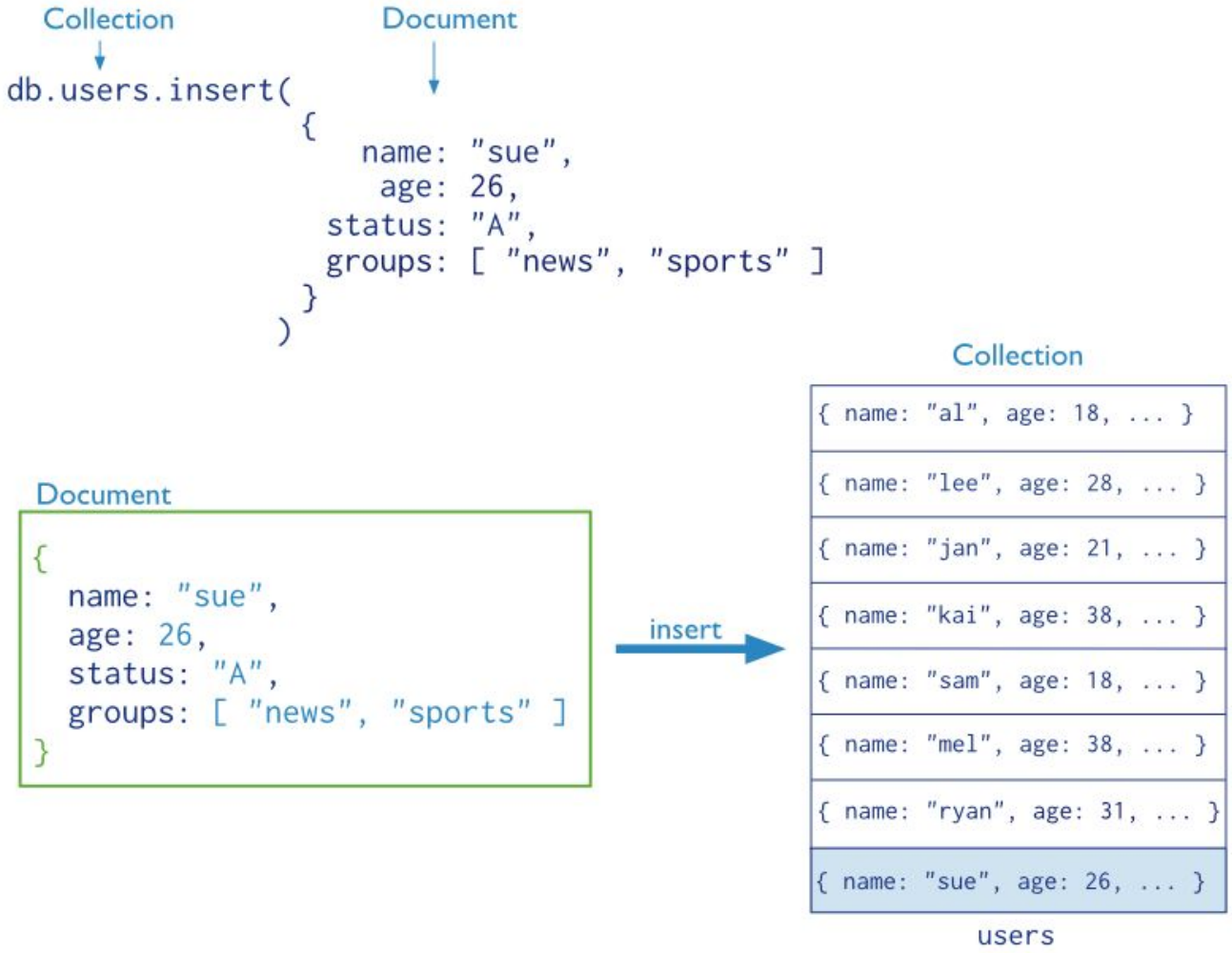
- Documents are independent units
- Application logic is easier to write
- Unstructured data can be stored easily

Document


{
name: "sue",
age: 26,
status: "A",
groups: ["news", "sports"]
}
field: value
field: value
field: value
field: value

Mongo - Add document to table

```
var db = client.db("test");
var collection = db.collection('documents');

var document = {
      name :'peter',
      email:'peter@mit.du',
      age  :'18'
};

// add document to database
collection.insertOne(document, function(err, res) {
   if (err) throw err;
   console.log("1 document inserted");
});
```


Collection
Document
db.users.insert(
{
name: "sue",
age: 26,
status: "A",
groups: ["news", "sports"]
}
)
Collection
{ name: "al", age: 18, ... }
{ name: "lee", age: 28, ... }
{ name: "jan", age: 21, ... }
{ name: "kai", age: 38, ... }
{ name: "sam", age: 18, ... }
{ name: "mel", age: 38, ... }
{ name: "ryan", age: 31, ... }
{ name: "sue", age: 26, ... }
users
Document
{
name: "sue",
age: 26,
status: "A",
groups: ["news", "sports"]
}
insert

Collection

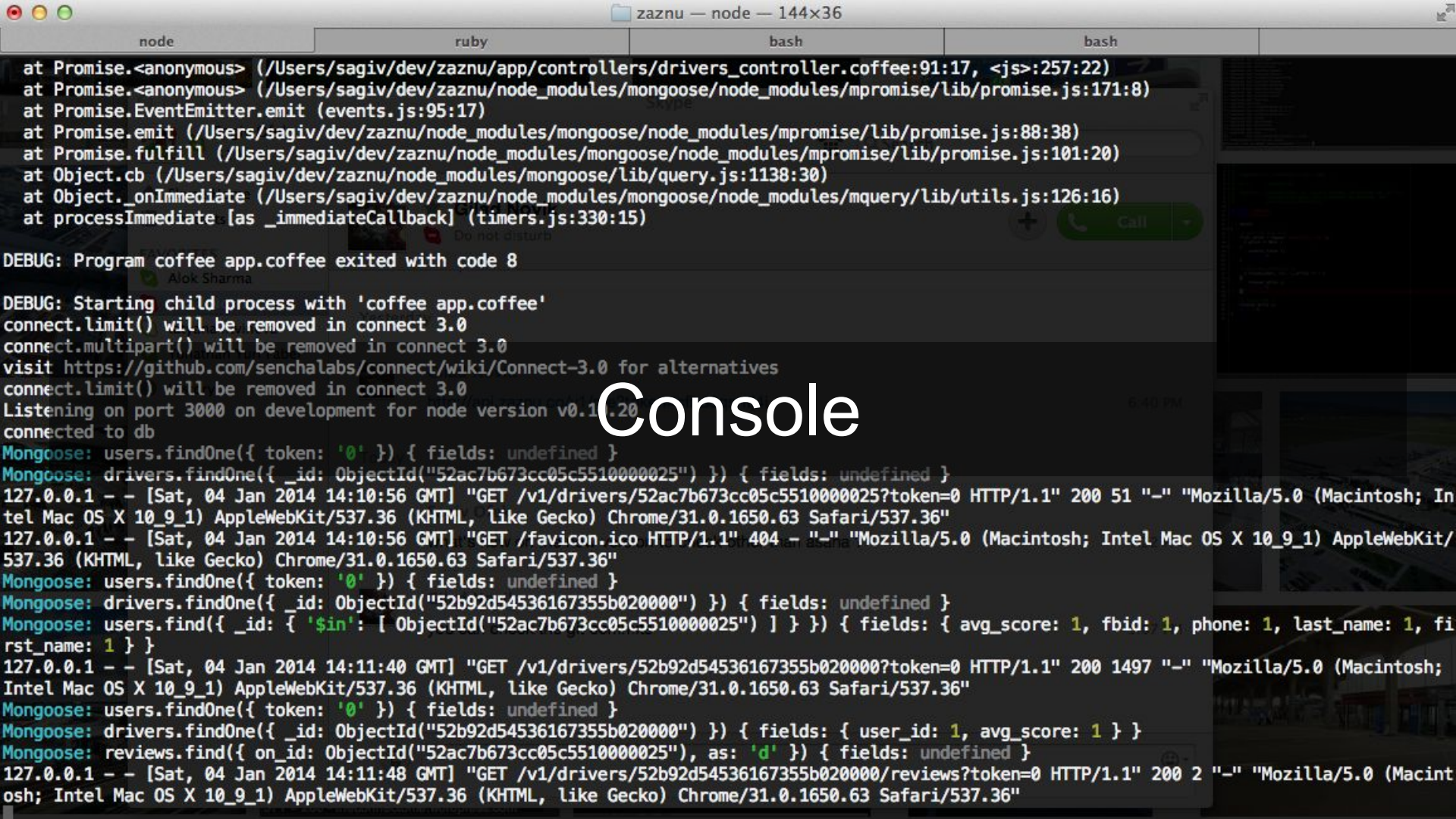
Multiple documents make a collection

Create collection

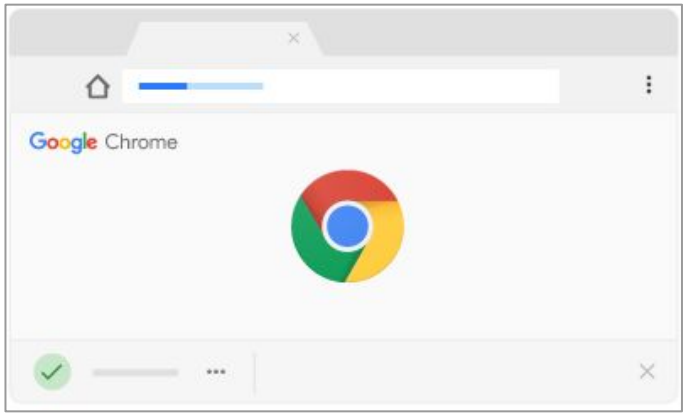
```
// get handle on database
var db = client.db("test");

// create documents table
var collection = db.collection('documents');
```

Read collection

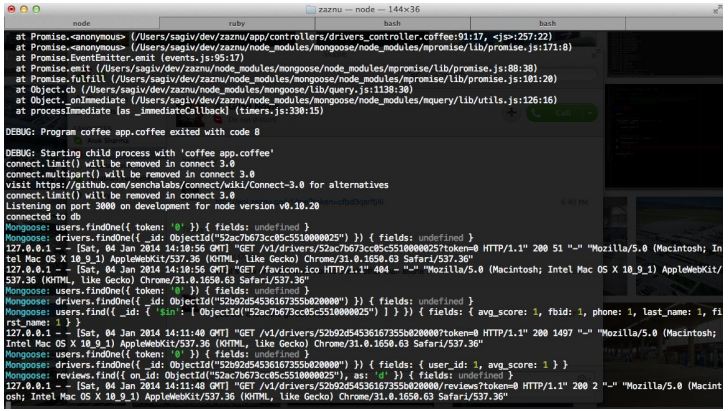
```
var db = client.db("test");
var collection = db.collection('documents');

// read database collection
collection.find({}).toArray(function(err, docs) {
   console.log("Found the following records");
   console.log(docs)
});
```

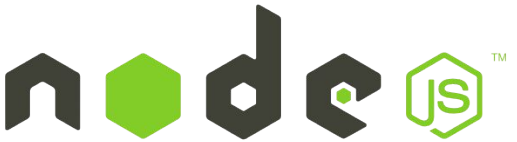

zaznu — node — 144×36
node
ruby
bash
bash
at Promise.<anonymous> (/Users/sagiv/dev/zaznu/app/controllers/drivers_controller.coffee:91:17, <js>:257:22)
at Promise.<anonymous> (/Users/sagiv/dev/zaznu/node_modules/mongoose/node_modules/mpromise/lib/promise.js:171:8)
at Promise.EventEmitter.emit (events.js:95:17)
at Promise.emit (/Users/sagiv/dev/zaznu/node_modules/mongoose/node_modules/mpromise/lib/promise.js:88:38)
at Promise.fulfill (/Users/sagiv/dev/zaznu/node_modules/mongoose/node_modules/mpromise/lib/promise.js:101:20)
at Object.cb (/Users/sagiv/dev/zaznu/node_modules/mongoose/lib/query.js:1138:30)
at Object._onImmediate (/Users/sagiv/dev/zaznu/node_modules/mongoose/node_modules/mquery/lib/utils.js:126:16)
at processImmediate [as _immediateCallback] (timers.js:330:15)
DEBUG: Program coffee app.coffee exited with code 8
DEBUG: Starting child process with 'coffee app.coffee'
connect.limit() will be removed in connect 3.0
connect.multipart() will be removed in connect 3.0
visit https://github.com/senchalabs/connect/wiki/Connect-3.0 for alternatives
connect.limit() will be removed in connect 3.0
Listening on port 3000 on development for node version v0.10.20
connected to db
Mongoose: users.findOne({ token: '0' }) { fields: undefined }
Mongoose: drivers.findOne({ _id: ObjectId("52ac7b673cc05c5510000025") }) { fields: undefined }
127.0.0.1 - - [Sat, 04 Jan 2014 14:10:56 GMT] "GET /v1/drivers/52ac7b673cc05c5510000025?token=0 HTTP/1.1" 200 51 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36"
127.0.0.1 - - [Sat, 04 Jan 2014 14:10:56 GMT] "GET /favicon.ico HTTP/1.1" 404 - "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36"
Mongoose: users.findOne({ token: '0' }) { fields: undefined }
Mongoose: drivers.findOne({ _id: ObjectId("52b92d54536167355b020000") }) { fields: undefined }
Mongoose: users.find({ _id: { '$in': [ObjectId("52ac7b673cc05c5510000025")] } }) { fields: { avg_score: 1, fbid: 1, phone: 1, last_name: 1, first_name: 1 } }
127.0.0.1 - - [Sat, 04 Jan 2014 14:11:40 GMT] "GET /v1/drivers/52b92d54536167355b020000?token=0 HTTP/1.1" 200 1497 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36"
Mongoose: users.findOne({ token: '0' }) { fields: undefined }
Mongoose: drivers.findOne({ _id: ObjectId("52b92d54536167355b020000") }) { fields: { user_id: 1, avg_score: 1 } }
Mongoose: reviews.find({ on_id: ObjectId("52ac7b673cc05c5510000025"), as: 'd' }) { fields: undefined }
127.0.0.1 - - [Sat, 04 Jan 2014 14:11:48 GMT] "GET /v1/drivers/52b92d54536167355b020000/reviews?token=0 HTTP/1.1" 200 2 "-" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_1) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/31.0.1650.63 Safari/537.36"
Console

Where is your data? In the browser? In the file system?

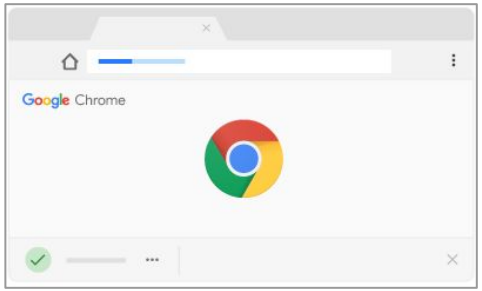


Browser

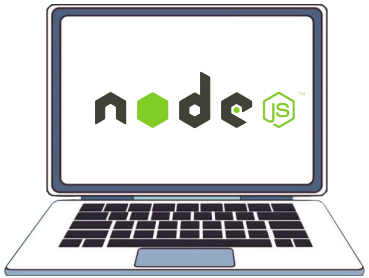




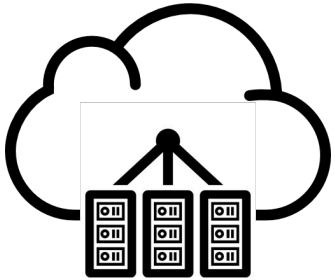
Where is your data?



Browser



Laptop



Cloud

Store data in memory

```
var express = require('express');
var app     = express();
var store   = [];

app.get('/add/:first/:last', function(req,res){
    var first = req.params.first;
    var last  = req.params.last;
    var item  = {first,last};
    store.push(item);
    res.send(item);
});
```

FAKER

```
> npm i faker
```

Generate massive fake data

```
var faker = require('faker');

// generate name
var randomName = faker.name.findName();

// generate email
var randomEmail = faker.internet.email();

// generate credit card
var randomCard = faker.helpers.createCard();
```

Store fake data

```
app.get('/add/:length', function(req,res){
    var length = req.params.length;
    for (var i = 0; i < length; i++) {
        var first = faker.name.firstName();
        var last  = faker.name.lastName();
        store.push({first,last});
    }
    res.send('added ' + length + ' users');
});
```

```
> npm i lowdb
```

lowdb - a little database

```
var low      = require('lowdb')
var FileSync = require('lowdb/adapters/FileSync')
var adapter  = new FileSync('db.json')
var db       = low(adapter)


db.get('posts')
 .push({ id: 1, title: 'lowdb is awesome'})
 .write()
```

Store bus locations

```
locations.forEach(function(bus){
   var id           = bus.id;
   var label        = bus.attributes.label
   var direction_id = bus.attributes.direction_id
   var latitude     = bus.attributes.latitude
   var longitude    = bus.attributes.longitude
   db.get('vehicles')
       .push({id,label,direction_id,latitude,longitude})
       .write()
});
```

Unique bus ids

```
var vehicles = db.get('vehicles').value()
var result = new Set(vehicles.map(vehicle => vehicle.id));


console.log(result);
console.log(Array.from(result));
```

Browser and Console

Data Model

- Structured:
  - Tables with predefined fields
  - Example: Browser IndexedDB
- Key/Value:
  - Key/Value stores, and NoSQL DBs
  - Example: Browser Cache API and Apache Cassandra
- Byte Streams:
  - Opaque string of bytes
  - Example: file systems and cloud storage services

Persistence

- Session persistence:
  - Lives as long as web session - until you close the tab
  - Example: Session Storage API
- Device persistence:
  - Persists across sessions - data is there next time you open browser
  - Example: Cache API
- Global persistence:
  - Persists across sessions & devices - data is there across devices
  - Example: AWS cloud storage

Write/Block

- Transactions:
  - Example: Atomicity, Consistency, Isolation, Durability - ACID
- Sync/Async:
  - Example: Blocking or parallel

Comparison - Browser APIs

API	Data Model	Persistence	Browser Support	Transactions	Sync/Async
File system	Byte stream	device	52%	No	Async
Local Storage	key/value	device	93%	No	Sync
Session Storage	key/value	session	93%	No	Sync
Cookies	structured	device	100%	No	Sync
WebSQL	structured	device	77%	Yes	Async
Cache	key/value	device	60%	No	Async
IndexedDB	hybrid	device	83%	Yes	Async
cloud storage	byte stream	global	100%	No	Both

Recommendations

- Offline storage: Cache API
- Offline storage: IndexedDB - broader browser support
- Global byte stream storage: Cloud Storage service.

Popular databases for Node JS

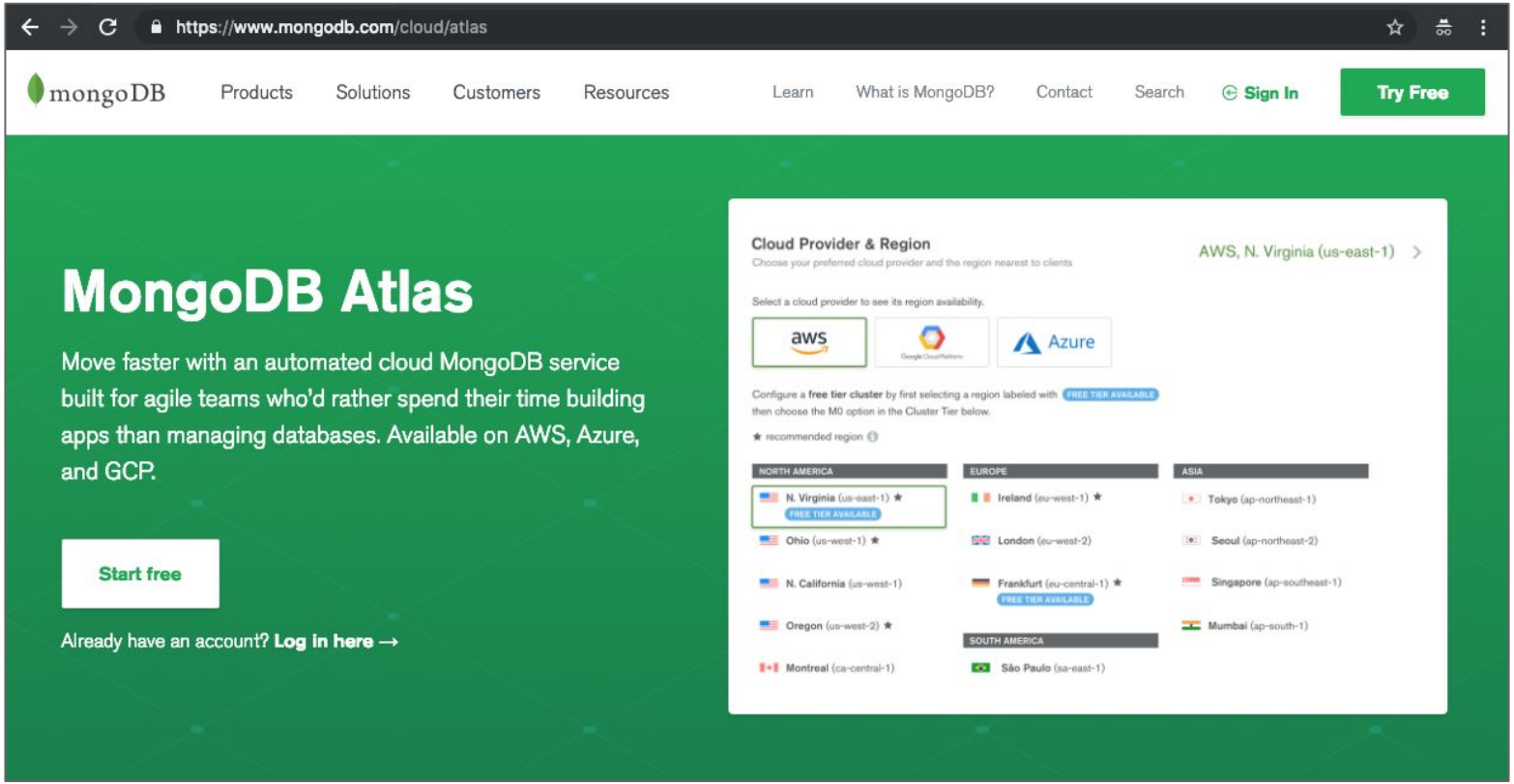
- MongoDB
  - https://www.mongodb.com/
- CouchDB
  - https://couchdb.apache.org/
- Redis
  - https://redis.io/

- Others
  - Cassandra
  - Couchbase
  - CouchDB
  - LevelDB
  - MySQL
  - MongoDB
  - Neo4j
  - Oracle
  - PostgreSQL
  - Redis
  - SQL Server
  - SQLite
  - ElasticSearch

* https://expressjs.com/en/guide/database-integration.html


Cloud

Sample free service - 512 MB



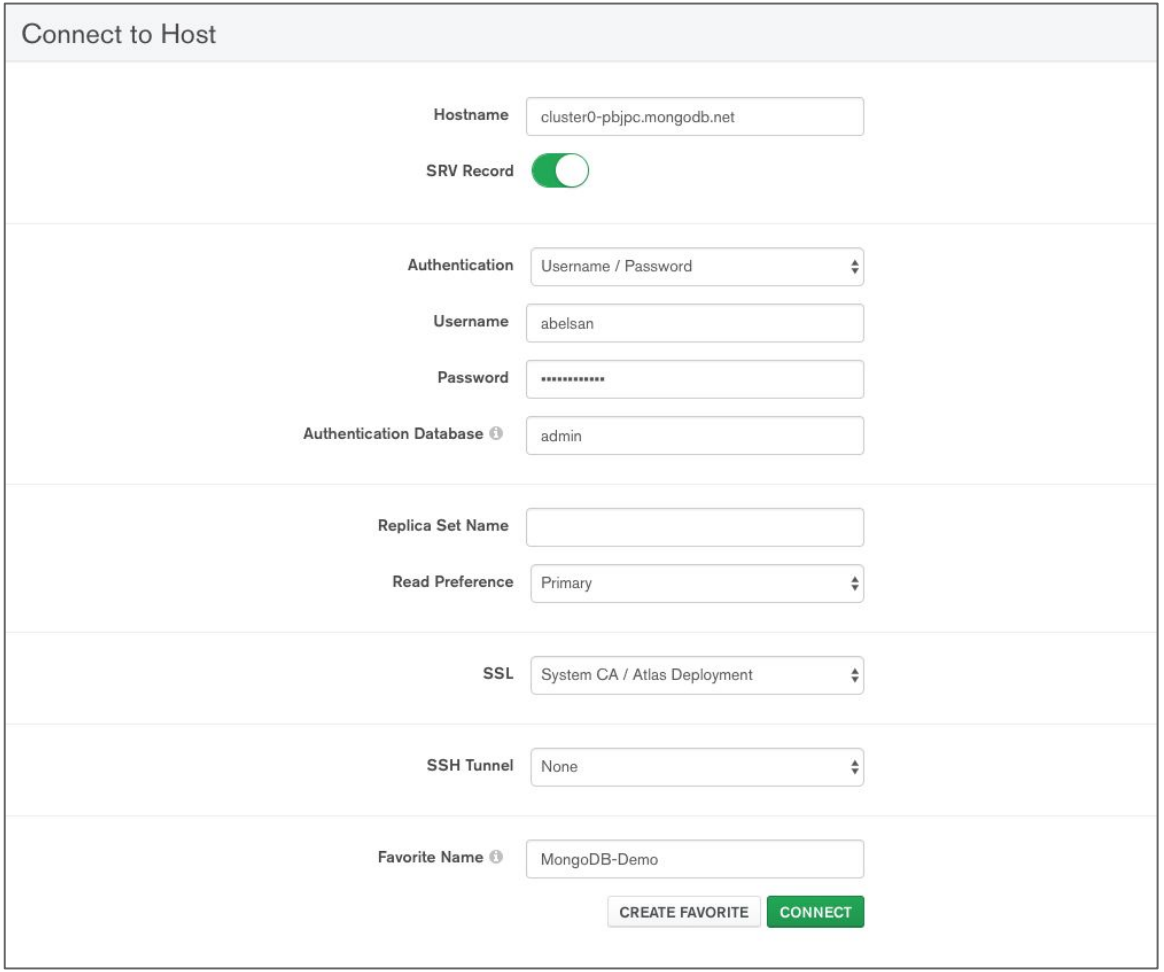
Setup Steps

- Build cluster
- Create user
- Whitelist your IP
- Connect to cluster

Mongo - Full example

```
var MongoClient = require('mongodb').MongoClient;
var uri = "mongodb+srv://<UserName>:<Password>@cluster0-pbjpc.mongodb.net/test?retryWrites=true";
var client = new MongoClient(uri, { useNewUrlParser: true });

client.connect(err => {
    var db = client.db("test");
    var collection = db.collection('documents');
    var document = {
            name :'peter',
            email:'peter@mit.du',
            age  :'18'
    };
    collection.insertOne(document, function(err, res) {
        if (err) throw err;
        console.log("1 document inserted");
    });
    client.close();
});
```

Compass


Connect to Host
Hostname
cluster0-pbjpc.mongodb.net
SRV Record
Authentication
Username / Password
Username
abelsan
Password
Authentication Database
admin
Replica Set Name
Read Preference
Primary
SSL
System CA / Atlas Deployment
SSH Tunnel
None
Favorite Name
MongoDB-Demo
CREATE FAVORITE
CONNECT