# COMMUNICATE WITH DATABASE

## Agenda

 See code samples for connecting to popular databases

### **SQLite**

- Self contained
- □ Full featured SQL
- ACID
- Zero configuration
- Single cross platform disk file
- Supports terabyte-sized databases
- □ One writer at a time ⊗

#### When to use

- Application file format
- Website database (small websites)
- □ Demo/Mock purposes

# Getting Started

#### yarn add sqlite3

```
const {Database} = require("sqlite3");
const {promisify} = require("util");
Database.prototype.all_P = promisify(Database.prototype.all);
main();
async function main() {
  const db = await connect("1.db");
  const rows = await db.all_P("SELECT * from contacts");
  console.log(rows);
function connect(filePath) {
  return new Promise((resolve, reject)=> {
    const db = new Database("1.db", err => {
       if(err) {
         reject(err);
         return;
       resolve(db);
    });
  });
```

## Inserting

```
function run(db, sql, args) {
  return new Promise((resolve,reject)=> {
    db.run(sql, args, function (err) {
       if (err) {
         reject(err);
         return;
       resolve(this);
    });
                      async function main() {
  });
                        const db = await connect("1.db");
                         const {lastID: id} = await run(db, "INSERT
                      INTO contacts(name) VALUES(?)", ["Tommy"]);
                        console.log(id);
```

## Classical SQL Databases

- By order of popularity
  - Oracle
  - MySQL
  - Microsoft SQL Server
  - PostgreSQL
  - DB2

□ Boring ...

## Scalability of RDBMS

- You can hardly scale it out
- Scaling-out reads (selects) is very difficult
- Scaling-out writes (between machines) is nearly impossible
- This is due to the ACID property of RDBMS transactions
- Enter the CAP theorem...

#### The CAP Theorem

- In a distributed/partitioned application, you can only pick two of the following
- Consistency
- Availability
- Partition Tolerance

#### BASE

- Basically Available
- □ **S**oft State
- Eventual Consistency
- It means that when an update is made in one replica, the other partitions will see it over time (there will be an inconsistency window)

#### NoSQL Databases

- By order of popularity
  - MongoDB Document store
  - ElasticSearch Search engine
  - Redis Key-value store
  - Cassandra Wide columns store

## MongoDB

- NoSQL document-based open-source DB
- Developed and supported by 10gen
- MongoDB is written in C++
- □ The name originated from the word: "humongous"
- Is used in production at: Disney, FourSquare, SAP,
   SourceForge, NY Times and many more

#### Main Features

- □ Documents are stored in BSON
- Can create an index for any document attribute
- High-availability by replication
- Automated Sharding
- Rich querying syntax
- In-place updates

# Getting Started

- Download from mongodb web site
- From the terminal

bin\mongod.exe --dbpath f:\mongodb\data

Play with the shell

bin\mongo.exe

- □ use mydb
- db.contacts.insert({name: "Ori"})
- db.contacts.find()

#### From NodeJS

#### yarn add mongodb

```
const {MongoClient} = require('mongodb');
main();
async function main() {
  const client = await MongoClient.connect('mongodb://localhost:27017');
  const db = client.db("mydb");
  const contacts = db.collection("contacts");
  await contacts.insertMany([{name: "Gaila"}]);
  const docs = await contacts.find({name: "Gaila"}).toArray();
  for(const doc of docs) {
    console.log(doc);
  client.close();
```

## Interesting Aspects

- Create index
- findAndModify
- Bulk write
- □ Read concern

#### Cloud DB

- MongoDB Atlas
- Amazon DynamoDB
- Microsoft Azure SQL Database
- Firebird
- □ Google BigQuery

## Summary

- No single winning database
- Probably we need use multiple databases
- Very easy to connect from Node.js