
epam
MongoDB
Authentication
Vadym Makhonin
March 2018

# AGENDA OF THE LECTURE

- SQL vs NoSQL
- MongoDB
- Mongoose
- Authentication
- Passport.js

# SQL VS NOSQL

# RELATIONAL DATABASE PROBLEMS

- Scalability
- Flexibility

**NoSQL** databases solve these problems

# NOSQL DATABASE PROBLEMS

- No join
- No data integrity
- No transaction

# WHERE SQL IS IDEAL

- logical related discrete data requirements which can be identified up-front
- data integrity is essential
- standards-based proven technology with good developer experience and support

# WHERE NOSQL IS IDEAL

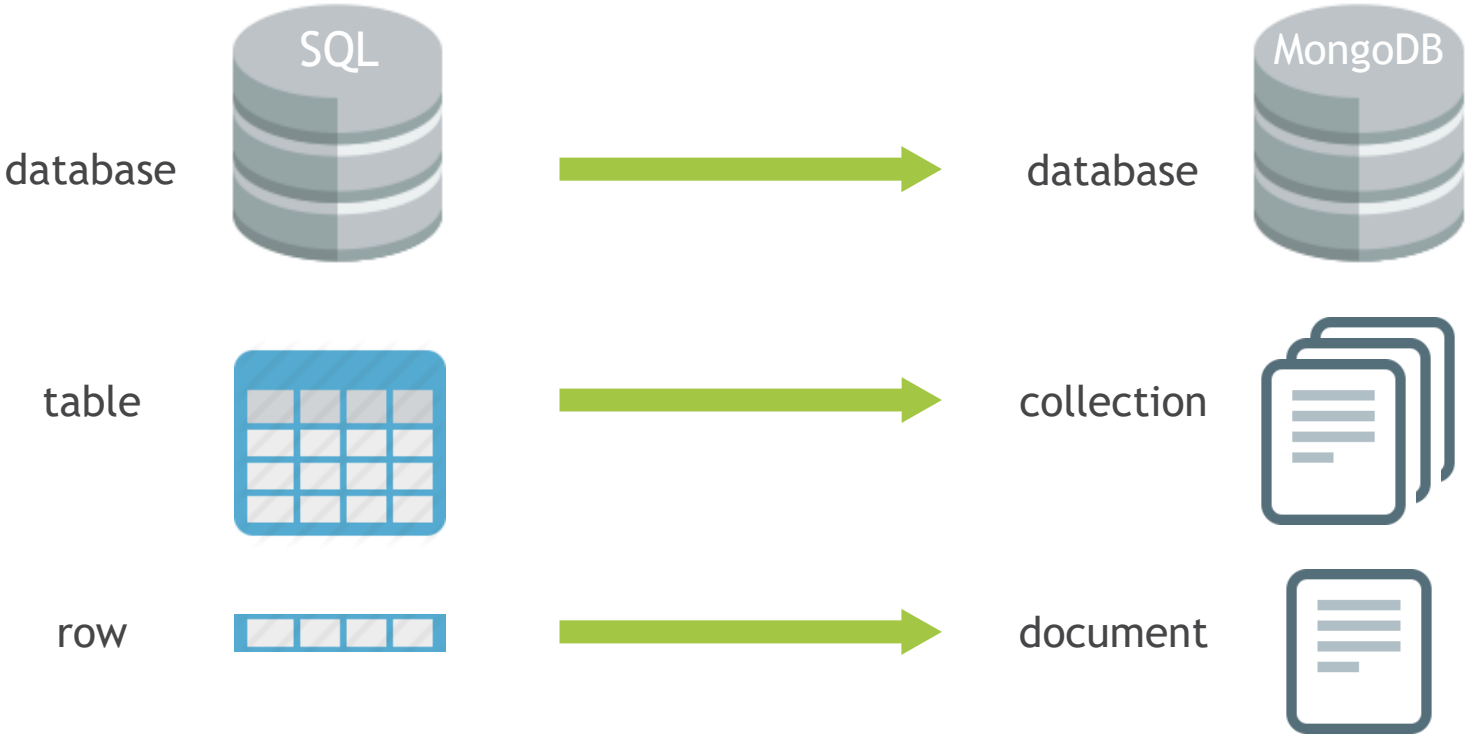
- unrelated, indeterminate or evolving data requirements
- simpler or looser project objectives, able to start coding immediately
- speed and scalability is imperative

MONGODB

# MONGODB


**MongoDB** is an open source, document-oriented database designed with both scalability and developer agility in mind.

# MONGODB COMPRASION TO SQL



# MOGNODB CLI

DEMO

# MONGODB CLI

- show dbs
- use <DB_NAME>
- show collections
- help / db.help() /db.collection.help()

# MONGODB CLI CRUD

- db.collection.insert(document)
- db.collection.find(query, projection)
- db.collection.update(query, update, options)
- db.collection.remove(query, options)

# MONGODB DRIVERS

An application communicates with MongoDB by way of a client library, called a **driver**, that handles all interaction with the database in a language appropriate to the application.

```
npm install mongodb
```

# MOGNODB NATIVE DRIVER

DEMO

# ORM, ODM

**ORM** (Object-Relational Mapping), **ODM** (Object Document Mapper) - programming technique for converting data between incompatible type systems in databases and object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language.

**ORM** – for relational databases, **ODM** – for NoSQL databases.

Most popular ORM in Node.js – **Sequelize**.

# MONGOOSE

# MONGOOSE


**Mongoose** provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.

# DEMO

# MONGOOSE API

- mongoose.connect(url, options)
- mongoose.Promise
- mongoose.Schema
- mongoose.model(name, schema)
- mongoose.plugin(func, options)

# SCHEMA API

```
const schema = new Schema(definition, options)
```

- schema.methods
- schema.statics
- schema.virtual(name, options)
- schema.pre/post(method, callback)
- schema.plugin(func, options)

# SCHEMA DEFINITION

- type
- required
- default
- unique
- validate

String

- lowercase
- uppercase
- trim
- match
- enum

Number/Date

- min
- max

# AUTHENTICATION

# AUTHENTICATION

- **Authentication** - is the process of actually confirming truth identity.
- **Authorization** - is the function of specifying access rights to resources related to information security and computer security in general and to access control in particular.

# AUTHENTICATION METHODS

- HTTP
- Forms
- One-Time Password(two-factor authentication)
- API key
- Token-based

# HOW TOKEN BASED WORKS

- User Requests Access with Username / Password
- Application validates credentials
- Application provides a signed token to the client
- Client stores that token and sends it along with every request
- Server verifies token and responds with data

# TOKEN-BASED AUTHENTICATION

Token formats:

- SWT
- JWT
- SAML

Standards:

- OAuth
- OpenID Connect
- SAML
- WS-Federation

# PASSPORT.JS

# PASSPORT

**Passport** is Express-compatible authentication middleware for Node.js.
**Passport's** sole purpose is to authenticate requests, which it does through an extensible set of plugins known as **strategies**. The API is simple: you provide **Passport** a request to authenticate, and **Passport** provides hooks for controlling what occurs when authentication succeeds or fails.

# PASSPORT MAIN CONCEPTS

- Strategies
- Sessions
- Middleware

# PASSPORT API

- passport.initialize / session()
- passport.use()
- passport.serializeUser / deserializeUser()
- passport.authenticate()
- req.login / logout()

# DEMO