# Sequelize Fast-Track Roadmap — Phased Lessons

**Goal:** Learn Sequelize (Node.js ORM) quickly and deeply — one topic at a time, with theory + analogy + practical examples + exercises.

## Prerequisites (what you should already know)

* Basic JavaScript (ES6+), async/await
* Node.js and npm/yarn
* Basic SQL (SELECT, JOIN, INSERT, UPDATE) — not deep, just concepts
* Familiarity with Express.js (for building APIs) and React (for front-end integration)
* Git and terminal comfort

# PHASE 0 — Quick Setup (must-have)

* Choose a relational DB: **Postgres (recommended)**, MySQL, MariaDB, SQLite (good for tests/dev).
* Packages you’ll typically use:
  + sequelize (core)
  + sequelize-cli (optional but highly recommended for migrations/seeds)
  + dialect driver: pg + pg-hstore (Postgres), mysql2 (MySQL), mariadb (MariaDB), sqlite3 (SQLite), tedious (Microsoft SQL Server) and oracledb (Oracle Database)

Quick CLI starter commands (cheat-sheet):

npm init -y  
npm install sequelize  
npm install --save-dev sequelize-cli  
# One of the following:  
$ npm install --save pg pg-hstore # Postgres  
$ npm install --save mysql2 # MySQL  
$ npm install --save mariadb # MariaDB  
$ npm install --save sqlite3 # SQLite  
$ npm install --save tedious # Microsoft SQL Server  
$ npm install --save oracledb # Oracle Database

# PHASE 1 — **Fundamentals** (minimum to be productive)

1. **What is Sequelize & when to use it**
   * Short: An ORM that maps JS objects to SQL tables and provides a high-level API.
   * Why it helps: Faster development, safer queries, cross-dialect portability.
   * Analogy: Sequelize is the translator between your JS code and the database.
2. **Project setup & connection**
   * Create Sequelize instance, environment config (dotenv), connection testing (sequelize.authenticate()), pool options.
   * Minimal code snippet included in lesson.
3. **Models & DataTypes**
   * sequelize.define() vs class Model extends Model + init().
   * DataTypes: STRING, INTEGER, BOOLEAN, DATE, JSON, TEXT, DECIMAL etc.
   * Field options: allowNull, defaultValue, unique, validate.
4. **Migrations (why & how)**
   * sequelize-cli setup, model:generate, migration up/down, running db:migrate.
   * Why migrations are preferable to sync({ force: true }) in production.
5. **CRUD basics**
   * create, findOne, findAll, findByPk, update, destroy.
   * findOrCreate, increment, decrement.
6. **Associations (basic)**
   * hasOne, belongsTo, hasMany, belongsToMany (through table).
   * FK ownership, onDelete/onUpdate behaviors.
7. **Querying & Operators**
   * where clause, Op operators (Op.gt, Op.like, Op.in, Op.or), attributes, order, limit, offset.
8. **Hooks & Validations**
   * Lifecycle hooks: beforeCreate, afterUpdate, etc.
   * Built-in validations and custom validators.
9. **Transactions (essential)**
   * Managed vs unmanaged transactions, passing { transaction: t }, rollback behavior.
10. **Integrating with Express (simple REST)**

* Pattern for controllers, error handling, request → DB flow.

# PHASE 2 — **Intermediate** (deeper practical skills)

1. **Advanced Associations**
   * Many-to-many through models with extra fields, aliasing (as), through options.
2. **Eager loading patterns**
   * Nested include, selecting attributes per association, required vs optional join, separate: true for large collections.
3. **Scopes & Query Helpers**
   * defaultScope, named scopes, reusable query patterns.
4. **Model options & indexes**
   * paranoid (soft delete), timestamps, underscored, schema support, indexes for performance.
5. **Bulk operations & performance**
   * bulkCreate, bulkUpdate (via update with where), upsert, RETURNING behavior.
6. **Raw queries & SQL security**
   * sequelize.query() with replacements/binds, avoiding SQL injection, when to use raw SQL.
7. **Pagination (offset & cursor-based)**
   * Implementing efficient pagination and considerations for large datasets.
8. **Connection pooling & config tuning**
   * Pool params, reconnect logic, logging control.
9. **Testing models**
   * In-memory SQLite for unit tests, factories, seeding test data, mocking.

# PHASE 3 — **Advanced**

These are advanced topics you can learn after the intermediate set.

1. **Polymorphic & Self-referential associations**
   * Implementing tagging systems, comment threading, recursive relations.
2. **Multi-tenant patterns**
   * Row-based vs schema-based tenancy, pros/cons, migration strategies.
3. **Zero-downtime migrations & production workflows**
   * Adding columns safely, backfilling data, rollouts.
4. **Complex query optimization**
   * Explain plans, index strategies, denormalization trade-offs.
5. **Sequelize + GraphQL + DataLoader**
   * N+1 problem, batching resolver patterns, dataloader integration.
6. **TypeScript + Sequelize**
   * Typings, sequelize-typescript or manual typing patterns, pros/cons.
7. **Custom data types, getters/setters, virtual fields**
   * Virtual attributes, JSON columns, custom casting.
8. **Contributing to Sequelize / reading source**
   * How to navigate the library codebase if you want to contribute or debug.

# Capstone Projects (pick one to build end-to-end)

* **Blog + Comments + Tags**: Users, Posts, Comments, Tags (many-to-many). Full REST API + React front-end. Auth, pagination, search.
* **E-commerce-ish**: Products, Categories, Orders, OrderItems, Inventory, Payments (mock). Multi-table transactions for checkout.
* **Job board**: Jobs, Companies, Applicants, resume upload (file handling), search filters.

Each capstone will be split into tasks and lessons (DB design, models, migrations, APIs, frontend integration, testing, deployment).

# Quick Best Practices (summary)

* Use migrations in all non-trivial projects; avoid sync({ force: true }) in prod.
* Keep models thin: validation + relations. Put business logic in services.
* Always use transactions when multiple related writes happen.
* Watch SQL logs while developing to understand generated queries.
* Use parameterized queries / replacements for raw SQL.
* Add indexes based on query patterns, not prematurely.

# How I will teach each topic (my lesson format)

1. One-paragraph explanation + real-world analogy
2. Minimal code example with comments (ready to run)
3. Step-by-step walkthrough of the code
4. Two small exercises (one easy, one slightly harder)
5. Answers & explanation
6. Common pitfalls & debugging tips