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**
2. Short: An ORM that maps JS objects to SQL tables and provides a high-level API.
3. Why it helps: Faster development, safer queries, cross-dialect portability.
4. Analogy: Sequelize is the translator between your JS code and the database.
5. **Project setup & connection**
6. Create Sequelize instance, environment config (dotenv), connection testing (`sequelize.authenticate()`), pool options.
7. Minimal code snippet included in lesson.
8. **Models & DataTypes**
9. `sequelize.define()` vs `class Model extends Model` + `init()`.
10. DataTypes: `STRING, INTEGER, BOOLEAN, DATE, JSON, TEXT, DECIMAL` etc.
11. Field options: `allowNull`, `defaultValue`, `unique`, `validate`.
12. **Migrations (why & how)**
13. `sequelize-cli` setup, `model:generate`, migration `up/down`, running `db:migrate`.
14. Why migrations are preferable to `sync({ force: true })` in production.
15. **CRUD basics**
16. `create`, `findOne`, `findAll`, `findByPk`, `update`, `destroy`.
17. `findOrCreate`, `increment`, `decrement`.
18. **Associations (basic)**
19. `hasOne`, `belongsTo`, `hasMany`, `belongsToMany` (through table).
20. FK ownership, `onDelete` / `onUpdate` behaviors.

21. **Querying & Operators**

22. `where` clause, `Op` operators (`Op.gt`, `Op.like`, `Op.in`, `Op.or`), `attributes`, `order`, `limit`, `offset`.

23. **Hooks & Validations**

24. Lifecycle hooks: `beforeCreate`, `afterUpdate`, etc.

25. Built-in validations and custom validators.

26. **Transactions (essential)**

27. Managed vs unmanaged transactions, passing `{ transaction: t }`, rollback behavior.

28. **Integrating with Express (simple REST)**

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

PHASE 2 — Intermediate (deeper practical skills)

1. **Advanced Associations**

2. Many-to-many through models with extra fields, aliasing (`as`), `through` options.

3. **Eager loading patterns**

4. Nested `include`, selecting attributes per association, `required` vs optional join, `separate: true` for large collections.

5. **Scopes & Query Helpers**

6. `defaultScope`, named scopes, reusable query patterns.

7. **Model options & indexes**

8. `paranoid` (soft delete), `timestamps`, `underscored`, schema support, indexes for performance.

9. **Bulk operations & performance**

10. `bulkCreate`, `bulkUpdate` (via `update` with where), `upsert`, `RETURNING` behavior.

11. **Raw queries & SQL security**

12. `sequelize.query()` with replacements/binds, avoiding SQL injection, when to use raw SQL.
13. **Pagination (offset & cursor-based)**
14. Implementing efficient pagination and considerations for large datasets.
15. **Connection pooling & config tuning**
16. Pool params, reconnect logic, logging control.
17. **Testing models**
18. 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**
2. Implementing tagging systems, comment threading, recursive relations.
3. **Multi-tenant patterns**
4. Row-based vs schema-based tenancy, pros/cons, migration strategies.
5. **Zero-downtime migrations & production workflows**
6. Adding columns safely, backfilling data, rollouts.
7. **Complex query optimization**
8. Explain plans, index strategies, denormalization trade-offs.
9. **Sequelize + GraphQL + DataLoader**
10. N+1 problem, batching resolver patterns, dataloader integration.
11. **TypeScript + Sequelize**
12. Typings, `sequelize-typescript` or manual typing patterns, pros/cons.
13. **Custom data types, getters/setters, virtual fields**

14. Virtual attributes, JSON columns, custom casting.

15. **Contributing to Sequelize / reading source**

16. 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