Reducing test execution time

(when using a database)

Mattis Turin-Zelenko, chax.at

Motivation

- Big project with many backend tests (using a database)
- Tests became slower over time
- …up to 2x45 minutes per build
- prevents devs from running tests locally
- clogging up build server

Outline

- Initial test setup - why were our tests slow?
- Improvement attempts - how can we make them faster?
- Transactional testing - how we sped up our tests by more than 80%!

Test Setup

- *(once)* Setup initial PostgreSQL database (DB) state
- For each test:
  - Clear database (`` `TRUNCATE TABLE table1, table2, ...` ``)
  - Setup test data (`` `INSERT INTO ...` ``)
  - Run test
- Database clearing is "slow"
  - ~500-1000ms
  - scales with project
- Result: 45min test execution time

Use in-memory SQLite database

- ORM allows us to swap underlying database to in-memory SQLite
- **Pro**
- …faster for tests than full PostgreSQL database
- …easy to set up when using an ORM
- **Con**
- …discrepancies between DB engines (e.g. `` `LIKE` `` case sensitivity), even when using an ORM (especially raw queries)
- …tests no longer test the real DB engine

Tune PostgreSQL configuration

- Non-durable PostgreSQL configuration speeds up DB engine
- Turn off fsync, synchronous commit, …
- **Pro**
- …easy to set up (just change config)
- **Con**
- …only marginal performance improvement
- Alternative PostgreSQL DB templates not really faster than `` `TRUNCATE` ``

Don’t reset DB before each test

- Only reset the DB once for each test suite/test file
- **Pro**
- …medium set up effort, lower if early in project
- …decent speed increase
- **Con**
- …developer must ensure tests work without reset
- …tests are no longer really independent

Mock

- Mock the database in tests
- **Pro**
- …fastest possible test execution time
- …great, if complicated functionality can be tested without DB
- **Con**
- …high set up effort
- …might mock away a lot of functionality
- …end-to-end tests probably still required (and slow)

Transactional testing

- Wrap each test case in a database transaction
- Transaction will be rolled-back after each test
- **Pro**
- …really fast
- …allows parallel test execution on the same DB
- …allows one-time seeding before test execution
- **Con**
- …requires some initial set up when not using a package

@chax-at/transactional-prisma-testing

- npm package `@chax-at/transactional-prisma-testing`
- allows you to automatically wrap tests in transactions, when using Prisma ORM
- handles most special cases
  - transactions in transactions (using `SAVEPOINT` in PostgreSQL)
  - parallel queries during test execution
  - …

Summary

- Truncating a DB for each test is slow, but required for clean state
- PostgreSQL configuration can be tuned for small improvements
  - Documentation: Non-Durable Settings
- DB calls can be mocked where possible
- Transactional testing can speed up tests
  - No test case changes required
  - `@chax-at/transactional-prisma-testing` npm package can handle set up