09

Faça como eu fiz: Desfazendo operações

Rodou o comando de migração antes de fazer alguma alteração importante em algum modelo e agora as tabelas do banco não estão como você precisa? Bom, já comentamos que as migrações em ORM também servem para termos um tipo de "versionamento" (feito através do arquivo `SequelizeMeta` no seu banco) e poder voltarmos o banco a um estado anterior à última alteração.

Como fazer isso? Através dos comandos:

```
npx sequelize-cli db:migrate:undo
```

Este comando vai desfazer somente a última migração feita, na ordem em que os arquivos são lidos e executados pelo Sequelize (de acordo com as datas e horários no nome dos arquivos). Se você tiver rodado 3 migrações - por exemplo, das tabelas `Niveis`, `Turmas` e `Matriculas`, o comando `npx sequelize-cli db:migrate:undo` vai desfazer apenas `Matriculas`.

Você pode rodar o mesmo comando novamente para ir desfazendo as migrações na ordem em que foram executadas, ou usar o comando:

```
db:migrate:undo --name [data-hora]-create-[nome-da-tabela].js
```

Para desfazer uma migração específica. Nesse caso, lembre-se que só irá funcionar se não tiver nenhuma outra tabela relacionada a ela!

Desfazendo seeds

Os seeds servem para termos dados iniciais no banco, normalmente dados de exemplo e/ou para teste. Quando você quiser desfazer essa operação para limpar esses dados do banco, pode rodar o comando:

```
npx sequelize db:seed:undo
```

Para desfazer o último seed feito.

```
npx sequelize-cli db:seed:undo --seed nome-do-arquivo
```

Para desfazer seeds de uma tabela específica.

```
npx sequelize-cli db:seed:undo:all
```

Para desfazer todos os seeds feitos.

Importante:

Ao contrário das migrações, não existe nenhum recurso de "versionamento" de seeds, só é possível incluir no banco e desfazer a operação (o que vai deletar os registros do banco).

Se você rodar o `:undo` em uma tabela e quiser mais tarde utilizar os seeds novamente na mesma tabela, os IDs deles serão outros. Por exemplo:

Rodamos o comando `npx sequelize-cli db:seed:all` e populamos a tabela:

id	nome	ativo	email	role	createdAt	updat
1	Ana Souza	1	ana@ana.com	estudante	2020-04-15 01:14:12	2020-0
2	Marcos Cintra	1	marcos@marcos.com	estudante	2020-04-15 01:14:12	2020-0
3	Felipe Cardoso	1	felipe@felipe.com	estudante	2020-04-15 01:14:12	2020-0
4	Sandra Gomes	0	sandra@sandra.com	estudante	2020-04-15 01:14:12	2020-0
5	Paula Morais	1	paula@paula.com	docente	2020-04-15 01:14:12	2020-0
6	Sergio Lopes	1	sergio@sergio.com	docente	2020-04-15 01:14:12	2020-0

Se rodarmos o comando `npx sequelize-cli db:seed:undo:all` para deletar esses registros e, em seguida, refazer os seed novamente com `npx sequelize-cli db:seed:all` , o resultado será o abaixo. Compare os números de `id` !

id	nome	ativo	email	role	createdAt	updat
7	Ana Souza	1	ana@ana.com	estudante	2020-04-15 01:14:12	2020-0
8	Marcos Cintra	1	marcos@marcos.com	estudante	2020-04-15 01:14:12	2020-0
9	Felipe Cardoso	1	felipe@felipe.com	estudante	2020-04-15 01:14:12	2020-0
10	Sandra Gomes	0	sandra@sandra.com	estudante	2020-04-15 01:14:12	2020-0
11	Paula Morais	1	paula@paula.com	docente	2020-04-15 01:14:12	2020-0
12	Sergio Lopes	1	sergio@sergio.com	docente	2020-04-15 01:14:12	2020-0

Os registros terão novos IDs, pois uma vez deletado o ID nunca é reutilizado. Se você estiver migrando/*seedando* tabelas relacionadas, é sempre bom conferir os IDs de todas, do contrário o Sequelize vai lançar um erro de relação.