Sequelize Cheatsheet

Command Line

Sequelize provides utilities for generating migrations, models, and seed files. They are exposed through the `sequelize-cli` command.

Init Project

```
$ npx sequelize-cli init
```

You must create a database user, and update the `config/config.json` file to match your database settings to complete the initialization process.

Create Database

```
$ npx sequelize-cli db:create
```

Generate a model and its migration

```
$ npx sequelize-cli model:generate --name <ModelName> --attributes <column1>:<type>,<column2>:<t
```

Run pending migrations

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

Rollback one migration

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

Rollback all migrations

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

Generate a new seed file

```
$ npx sequelize-cli seed:generate --name <descriptiveName>
```

Run all pending seeds

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

Rollback one seed

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

Rollback all seeds

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

Migrations

Column Attribute Keywords

```
<columnName>: {
    type: Sequelize.<type>,
    allowNull: <true|false>,
    unique: <true|false>,
    references: { model: <TableName> }, // This is the plural table name that the column referer
}
```

Model Associations

One to One between Student and Scholarship

student.js

```
    Student.hasOne(models.Scholarship, { foreignKey: 'studentId' });
```

scholarship.js

```
Scholarship.belongsTo(models.Student, { foreignKey: 'studentId' });
```

One to Many between Student and Class

student.js

```
Student.belongsTo(models.Class, { foreignKey: 'classId' });
```

class.js

```
Class.hasMany(models.Student, { foreignKey: 'classId' });
```

Many to Many between Student and Lesson through StudentLessons table

student.js

```
const columnMapping = {
    through: 'StudentLesson', // This is the model name referencing the join table.
    otherKey: 'lessonId',
    foreignKey: 'studentId'
}

Student.belongsToMany(models.Lesson, columnMapping);
```

lesson.js

```
const columnMapping = {
    through: 'StudentLesson', // This is the model name referencing the join table.
    otherKey: 'studentId',
    foreignKey: 'lessonId'
}

Lesson.belongsToMany(models.Student, columnMapping);
```

Query Format

findOne

```
<Model>.findOne({
  where: {
        <column>: {
                [Op.<operator>]: <value>
        }
  },
});
```

findAll

```
<Model>.findAll({
  where: {
    <column>: {
        [Op.<operator>]: <value>
    }
  },
  include: <include_specifier>,
  offset: 10,
  limit: 2
});
```

findByPk

```
<Model>.findByPk(<primary_key>, {
        include: <include_specifier>
});
```

Common Where Operators

```
const Op = Sequelize.Op

[Op.and]: [{a: 5}, {b: 6}] // (a = 5) AND (b = 6)
[Op.or]: [{a: 5}, {a: 6}]  // (a = 5 OR a = 6)
[Op.gt]: 6,                // > 6
[Op.gte]: 6,               // >= 6
[Op.lt]: 10,               // < 10
[Op.lte]: 10,              // <= 10
[Op.ne]: 20,               // != 20
[Op.eq]: 3,                // = 3
[Op.is]: null              // IS NULL
[Op.not]: true,            // IS NOT TRUE
[Op.between]: [6, 10],     // BETWEEN 6 AND 10
[Op.notBetween]: [11, 15], // NOT BETWEEN 11 AND 15
[Op.in]: [1, 2],           // IN [1, 2]
[Op.notIn]: [1, 2],        // NOT IN [1, 2]
[Op.like]: '%hat',         // LIKE '%hat'
[Op.notLike]: '%hat'       // NOT LIKE '%hat'
[Op.iLike]: '%hat'         // ILIKE '%hat' (case insensitive) (PG only)
[Op.notILike]: '%hat'      // NOT ILIKE '%hat'  (PG only)
[Op.startsWith]: 'hat'     // LIKE 'hat%'
[Op.endsWith]: 'hat'       // LIKE '%hat'
[Op.substring]: 'hat'      // LIKE '%hat%'
[Op.regexp]: '^[h|a|t]'    // REGEXP/~ '^[h|a|t]' (MySQL/PG only)
[Op.notRegexp]: '^[h|a|t]' // NOT REGEXP/!~ '^[h|a|t]' (MySQL/PG only)
[Op.iRegexp]: '^[h|a|t]'    // ~* '^[h|a|t]' (PG only)
[Op.notIRegexp]: '^[h|a|t]' // !~* '^[h|a|t]' (PG only)
[Op.like]: { [Op.any]: ['cat', 'hat']}
```