

Model Querying - Basics

Model Querying - Finders

Getters, Setters &amp; Virtuals

Validations &amp; Constraints

Raw Queries

Associations

Paranoid

Advanced Association  
Concepts

Eager Loading

Creating with Associations

Advanced M:N Associations

Association Scopes

Polymorphic Associations

Other topics

Other Data Types

Using sequelize in AWS Lambda

Connection Pool

Constraints &amp; Circularities

Dialect-Specific Things

Extending Data Types

Hooks

Indexes

Working with Legacy Tables

[Home](#) > [Other topics](#) > [Constraints & Circularities](#)

Version: v6 - stable

# Constraints & Circularities

Adding constraints between tables means that tables must be created in the database in a certain order, when using `sequelize.sync`. If `Task` has a reference to `User`, the `User` table must be created before the `Task` table can be created. This can sometimes lead to circular references, where Sequelize cannot find an order in which to sync. Imagine a scenario of documents and versions. A document can have multiple versions, and for convenience, a document has a reference to its current version.

```
const { Sequelize, Model, DataTypes } = require("sequelize");

class Document extends Model {}
Document.init({
  author: DataTypes.STRING
}, { sequelize, modelName: 'document' });

class Version extends Model {}
Version.init({
  timestamp: DataTypes.DATE
}, { sequelize, modelName: 'version' });

Document.hasMany(Version); // This adds documentId attribute to version
Document.belongsTo(Version, {
  as: 'Current',
  foreignKey: 'currentVersionId'
}); // This adds currentVersionId attribute to document
```

However, unfortunately the code above will result in the following error:

```
Cyclic dependency found. documents is dependent of itself. Dependency chain: documents -> versions => documents
```

In order to alleviate that, we can pass `constraints: false` to one of the associations:

```
Document.hasMany(Version);
Document.belongsTo(Version, {
  as: 'Current',
  foreignKey: 'currentVersionId',
  constraints: false
});
```

Which will allow us to sync the tables correctly:

```
CREATE TABLE IF NOT EXISTS "documents" (
  "id" SERIAL,
  "author" VARCHAR(255),
  "createdAt" TIMESTAMPT WITH TIME ZONE NOT NULL,
  "updatedAt" TIMESTAMPT WITH TIME ZONE NOT NULL,
  "currentVersionId" INTEGER,
  PRIMARY KEY ("id")
);

CREATE TABLE IF NOT EXISTS "versions" (
  "id" SERIAL,
  "timestamp" TIMESTAMPT WITH TIME ZONE,
  "createdAt" TIMESTAMPT WITH TIME ZONE NOT NULL,
  "updatedAt" TIMESTAMPT WITH TIME ZONE NOT NULL,
  "documentId" INTEGER REFERENCES "documents" ("id") ON DELETE
SET
NULL ON UPDATE CASCADE,
  PRIMARY KEY ("id")
);
```

## Enforcing a foreign key reference without constraints

Sometimes you may want to reference another table, without adding any constraints, or associations. In that case you can manually add the reference attributes to your schema definition, and mark the relations between them.

```
class Trainer extends Model {}
Trainer.init({
  firstName: Sequelize.STRING,
  lastName: Sequelize.STRING
}, { sequelize, modelName: 'trainer' });

// Series will have a trainerId = Trainer.id foreign reference key
// after we call Trainer.hasMany(series)
class Series extends Model {}
Series.init({
  title: Sequelize.STRING,
  subTitle: Sequelize.STRING,
  description: Sequelize.TEXT,
  // Set FK relationship (hasMany) with `Trainer`
  trainerId: {
    type: DataTypes.INTEGER,
    references: {
      model: Trainer,
      key: 'id'
    }
  }
}, { sequelize, modelName: 'series' });

// Video will have seriesId = Series.id foreign reference key
// after we call Series.hasOne(Video)
class Video extends Model {}
Video.init({
  title: Sequelize.STRING,
  sequence: Sequelize.INTEGER,
  description: Sequelize.TEXT,
  // set relationship (hasOne) with `Series`
  seriesId: {
```



Need a better way to test your web app? Testim's advanced AI automates UI testing. Try it for free.

ADS VIA CARBON

Enforcing a foreign key reference without constraints

```
type: DataTypes.INTEGER,
references: {
  model: Series, // Can be both a string representing the table name or a Sequelize model
  key: 'id'
}
}, { sequelize, modelName: 'video' });

Series.hasOne(Video);
Trainer.hasMany(Series);
```



[Edit this page](#)

Last updated on **Oct 13, 2022** by **Rik Smale**





Previous  
[« Connection Pool](#)

Next  
[Dialect-Specific Things »](#)


#### Docs

[Guides](#)  
[Version Policy](#)  
[Security](#)   
[Changelog](#) 

#### Community

[Stack Overflow](#)   
[Slack](#)   
[Twitter](#)   
[GitHub](#) 

#### Support

[OpenCollective](#) 

Copyright © 2022 Sequelize Contributors.  
Built with Docusaurus and powered by Netlify.