HINTS

CONSEJOS CONCEPTUALES

- Cuando se especifica la opción through, en la relación belongsToMany, Sequelize automáticamente crea la tabla unión, que en este caso es ActorPeliculas. En postgreSQL sería:

```
CREATE TABLE IF NOT EXISTS "ActorPeliculas" (
 "createdAt" TIMESTAMP WITH TIME ZONE NOT NULL,
 "updatedAt" TIMESTAMP WITH TIME ZONE NOT NULL,
 "MovieId" INTEGER REFERENCES "Movies" ("id")ON DELETE CASCADE ON
UPDATE CASCADE,
 "ActorId" INTEGER REFERENCES "Actors"        ("id") ON DELETE
CASCADE ON UPDATE CASCADE,
 PRIMARY KEY ("MovieId","ActorId") );
```

- A diferencia de las relaciones uno a uno, y uno a muchos, los valores predeterminados para ON UPDATE y ON DELETE son CASCADA para las relaciones muchos a muchos.
- En el ORM Sequelize, Belongs-To-Many crea una clave única en el modelo. Este nombre se puede anular mediante la opción uniqueKey. Y para evitar su creación, utilice la opción unique: false.
- Las relaciones en Sequelize son definidas en pares:
  - Para crear una relación Uno-a-Uno, las relaciones hasOne y belongsTo son usadas de manera conjunta.
  - Para crear una relación Uno-a-Muchos, las relaciones hasMany y belongsTo son usadas de manera conjunta.
  - Para crear una relación Muchos-a-Muchos, dos llamadas a belongsToMany son usadas de manera conjunta.

En Sequelize, cuando una relación es definida entre dos modelos, solo el de tipo fuente conoce acerca de la relación. Por ejemplo: cuando se utiliza Foo.hasOne(Bar), Foo es el modelo fuente, y Bar es el modelo objetivo. En este caso, solo el primero de ellos conoce acerca de la existencia de la relación.

Es por eso que, en este caso, las instancias de Foo obtienen los métodos: getBar(), setBar(), y createBar(), mientras que las instancias de Bar no obtienen nada.

- El truco para decidir entre sourceKey y targetKey, es simplemente recordar dónde

coloca cada relación su clave externa.

- A.belongsTo(B) mantiene la clave externa en el modelo de origen (A), por lo tanto, a la que se hace referencia está en el modelo de destino; de ahí el uso de targetKey.
- A.hasOne(B) y A.hasMany(B) mantienen la clave externa en el modelo de destino (B), por lo tanto, a la que se hace referencia está en el modelo de origen; de ahí el uso de sourceKey.
- A.belongsToMany(B) implica una tabla adicional (la tabla de unión), por lo que se pueden utilizar tanto sourceKey, como targetKey; con el primero correspondiente a algún campo en A (la fuente), y con el segundo correspondiente a algún campo en B (el destino).