**Связь «Многие-со-Многими»(6.1.4.3, 6.1.4.4, 6.1.5)**

/\*

По соглашению именования в Laravel, имена связываемых таблиц должно быть во множественном числе (mashines, spares); имя связующей таблицы в единственном числе и содержать имена связываемых таблиц в алфавитном порядке (mashine\_spare); наименование связующих колонок в связующей таблице в единственном числе(mashine\_id, spare\_id); Наименование моделей связываемых таблиц в единственном числе (Machine, Spare); наименование методов моделей связываемых таблиц во множественном числе(machines(), spares()).

\*/

**Миграция связываемой перввой таблицы:**

Schema::create('mashines', function (Blueprint $table) {

$table->id();

$table->string('nameM', 30);

$table->timestamps();

});

**Миграция связываемой второй таблицы:**

Schema::create('spares', function (Blueprint $table) {

$table->id();

$table->string('nameS', 30);

$table->timestamps();

});

**Миграция связующей таблицы:**

Schema::create('mashine\_spare', function (Blueprint $table) {

$table->foreignId('mashine\_id')->constrained()->cascadeOnDelete();

$table->foreignId('spare\_id')->constrained()->cascadeOnDelete();

});

/\*

foreignId() - создаст в связующей таблице колонки 'mashine\_id' и 'spare\_id'.

constrained() - "создаст из" foreignId() связьс колонок связующей таблицы: 'mashine\_id' с таблицей 'mashines' колонкой ' id'; 'spare\_id' с таблицей 'spares' колонкой ' id'.

\*/

**Модель связываемой перввой таблицы:**

use App\Models\Spare;

class Mashine extends Model{

protected $fillable = ['nameM']; - чтобы работало массовое присвоенме

public function spares(){

return $this->belongsToMany(Spare::class);

}

}

**Модель связываемой второй таблицы:**

use App\Models\Mashine;

class Spare extends Model{

protected $fillable = ['nameS'];

public function machines(){

return $this->belongsToMany(Mashine::class);

}

}

/\*

Внимание! Имена методов связи должны быть во множественном числе и совпадать с именами колонок связующей таблицы spares() = spare\_id, machines() = machine\_id.

\*/

**Artisan tinker:**

use App\Models\Mashine as M;

use App\Models\Spare as S;

//Создать запись в первой таблице методом create()

M::create(['nameM' => 'bmw']);

M::create(['nameM' => 'Audi']);

M::create(['nameM' => 'Toyota']);

//Создать объект первой модели

$objM = M::find(1);

//Сохранить записи во второй таблице и связать их с первой записью первой таблицы при помощи метода createMany()

$objM->spares()->createMany([['nameS' => 'wheel 20'], ['nameS' => 'wheel 22'], ['nameS' => 'wheel 24']]);

//Сохранить записи во второй таблице методом create()

$objS4 = S::create(['nameS' => 'wheel 27']);

$objS5 = S::create(['nameS' => 'wheel 32']);

//Создать еще один объект первой модели

$objM2 = M::find(2);

//Связать запись 2 первой таблицы и записи 4,5 второй таблицы через объект первой модели

$objM2->spares()->attach([$objS3->id, $objS4->id]);

**- Также поддерживает методы:** sync(), syncWithoutDetaching(), toggle(), save (), saveMany () см книгу.

**- И смотри похожий пример One to Many.**

**- Склеивание таблиц в запроссе или SQL Join**(7.5.3.)**:**

смотри One to Many