//->onDelete(**'cascade'**)

<https://laravel.com/docs/5.1/eloquent#soft-deleting>

### **Soft Deleting**

In addition to actually removing records from your database, Eloquent can also "soft delete" models. When models are soft deleted, they are not actually removed from your database. Instead, adeleted\_at attribute is set on the model and inserted into the database. If a model has a non-nulldeleted\_at value, the model has been soft deleted. To enable soft deletes for a model, use theIlluminate\Database\Eloquent\SoftDeletes trait on the model and add the deleted\_at column to your $dates property:

<?php

namespace App;

use Illuminate\Database\Eloquent\Model;

use Illuminate\Database\Eloquent\SoftDeletes;

class Flight extends Model

{

use SoftDeletes;

/\*\*

\* The attributes that should be mutated to dates.

\*

\* @var array

\*/

protected $dates = ['deleted\_at'];

}

Of course, you should add the deleted\_at column to your database table. The Laravel [schema builder](https://laravel.com/docs/5.1/migrations) contains a helper method to create this column:

Schema::table('flights', function ($table) {

$table->softDeletes();

});

Now, when you call the delete method on the model, the deleted\_at column will be set to the current date and time. And, when querying a model that uses soft deletes, the soft deleted models will automatically be excluded from all query results.

To determine if a given model instance has been soft deleted, use the trashed method:

if ($flight->trashed()) {

//

}

### **Querying Soft Deleted Models**

#### Including Soft Deleted Models

As noted above, soft deleted models will automatically be excluded from query results. However, you may force soft deleted models to appear in a result set using the withTrashed method on the query:

$flights = App\Flight::withTrashed()

->where('account\_id', 1)

->get();

The withTrashed method may also be used on a [relationship](https://laravel.com/docs/5.1/eloquent-relationships) query:

$flight->history()->withTrashed()->get();

#### Where Clause Caveats

When adding orWhere clauses to your queries on soft deleted models, always use [advance where clauses](http://laravel.com/docs/5.1/queries#advanced-where-clauses) to logically group the WHERE clauses. For example:

User::where(function($query) {

$query->where('name', '=', 'John')

->orWhere('votes', '>', 100);

})

->get();

This will produce the following SQL:

select \* from `users` where `users`.`deleted\_at` is null and (`name` = 'John' or `votes` > 100)

If the orWhere clause is not grouped, it will produce the following SQL which will contain soft deleted records:

select \* from `users` where `users`.`deleted\_at` is null and `name` = 'John' or `votes` > 100

#### Retrieving Only Soft Deleted Models

The onlyTrashed method will retrieve **only** soft deleted models:

$flights = App\Flight::onlyTrashed()

->where('airline\_id', 1)

->get();

#### Restoring Soft Deleted Models

Sometimes you may wish to "un-delete" a soft deleted model. To restore a soft deleted model into an active state, use the restore method on a model instance:

$flight->restore();

You may also use the restore method in a query to quickly restore multiple models:

App\Flight::withTrashed()

->where('airline\_id', 1)

->restore();

Like the withTrashed method, the restore method may also be used on [relationships](https://laravel.com/docs/5.1/eloquent-relationships):

$flight->history()->restore();

#### Permanently Deleting Models

Sometimes you may need to truly remove a model from your database. To permanently remove a soft deleted model from the database, use the forceDelete method:

// Force deleting a single model instance...

$flight->forceDelete();

// Force deleting all related models...

$flight->history()->forceDelete();