Using Eloquent Relationships

Let's create a sophisticated relationship example using four tables in a Laravel application. We'll set up a scenario involving authors, books, genres, and reviews:

Setup a Schema for the Relationships:

Open your MySQL database server and create the schema '**libretto-<account>**'. This will be the schema that will be used to contain the migrated tables and relationships

Explanation of Relationships:

- **authors to books**: One-to-Many relationship (an author can write many books, but each book belongs to one author).
- **books to reviews**: One-to-Many relationship (a book can have many reviews, but each review belongs to one book).
- **books to genres**: Many-to-Many relationship (a book can belong to many genres, and each genre can have many books).
- **book_genre**: A pivot table for the many-to-many relationship between books and genres.

Migration Files

Authors Table

```
// database/migrations/2024_06_10_000000_create_authors_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateAuthorsTable extends Migration
{
    public function up()
    {
        Schema::create('authors', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('authors');
    }
}
```

Books Table

```
// database/migrations/2024_06_10_000001_create_books_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateBooksTable extends Migration
{
    public function up()
    {
        Schema::create('books', function (Blueprint $table) {
            $table->id();
            $table->string('title');
            $table->foreignId('author_id')->constrained()-
>onDelete('cascade');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('books');
    }
}
```

Genres Table

```
// database/migrations/2024_06_10_000002_create_genres_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateGenresTable extends Migration
{
    public function up()
    {
        Schema::create('genres', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('genres');
    }
}
```

Book Genre Pivot Table

```
// database/migrations/2024_06_10_000003_create_book_genre_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateBookGenreTable extends Migration
{
    public function up()
    {
        Schema::create('book_genre', function (Blueprint $table) {
            $table->id();
            $table->foreignId('book_id')->constrained()->onDelete('cascade');
            $table->foreignId('genre_id')->constrained()-
>onDelete('cascade');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('book_genre');
    }
}
```

Reviews Table

```
// database/migrations/2024_06_10_000004_create_reviews_table.php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

class CreateReviewsTable extends Migration
{
    public function up()
    {
        Schema::create('reviews', function (Blueprint $table) {
            $table->id();
            $table->foreignId('book_id')->constrained()->onDelete('cascade');
            $table->text('content');
            $table->integer('rating');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::dropIfExists('reviews');
    }
}
```

Models

Author Model

```
// app/Models/Author.php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Author extends Model
{
    use HasFactory;

    protected $fillable = ['name'];

    public function books()
    {
        return $this->hasMany(Book::class);
    }
}
```

Book Model

```
// app/Models/Book.php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Book extends Model
{
    use HasFactory;

    protected $fillable = ['title', 'author_id'];

    public function author()
    {
        return $this->belongsTo(Author::class);
    }

    public function genres()
    {
        return $this->belongsToMany(Genre::class);
    }

    public function reviews()
    {
        return $this->hasMany(Review::class);
    }
}
```

Genre Model

```
// app/Models/Genre.php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Genre extends Model
{
    use HasFactory;

    protected $fillable = ['name'];

    public function books()
    {
        return $this->belongsToMany(Book::class);
    }
}
```

Review Model

```
// app/Models/Review.php
namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;

class Review extends Model
{
    use HasFactory;

    protected $fillable = ['book_id', 'content', 'rating'];

    public function book()
    {
        return $this->belongsTo(Book::class);
    }
}
```

Factories

This part will create the factories that will mock the data that will be populating your tables. Use "`php artisan make:factory <Factory Name>`" in order to do this.

Author Factory

```
// database/factories/AuthorFactory.php
namespace Database\Factories;
```

```
use App\Models\Author;
use Illuminate\Database\Eloquent\Factories\Factory;

class AuthorFactory extends Factory
{
    protected $model = Author::class;

    public function definition()
    {
        return [
            'name' => $this->faker->name,
        ];
    }
}
```

Book Factory

```
// database/factories/BookFactory.php
namespace Database\Factories;

use App\Models\Book;
use App\Models\Author;
use Illuminate\Database\Eloquent\Factories\Factory;

class BookFactory extends Factory
{
    protected $model = Book::class;

    public function definition()
    {
        return [
            'title' => $this->faker->sentence,
            'author_id' => Author::factory(),
        ];
    }
}
```

Genre Factory

```
// database/factories/GenreFactory.php
namespace Database\Factories;

use App\Models\Genre;
use Illuminate\Database\Eloquent\Factories\Factory;

class GenreFactory extends Factory
{
    protected $model = Genre::class;

    public function definition()
```

```
    {
        return [
            'name' => $this->faker->word,
        ];
    }
}
```

Review Factory

```
// database/factories/ReviewFactory.php
namespace Database\Factories;

use App\Models\Review;
use App\Models\Book;
use Illuminate\Database\Eloquent\Factories\Factory;

class ReviewFactory extends Factory
{
    protected $model = Review::class;

    public function definition()
    {
        return [
            'book_id' => Book::factory(),
            'content' => $this->faker->paragraph,
            'rating' => $this->faker->numberBetween(1, 5),
        ];
    }
}
```

Seeder

```
// database/seeders/DatabaseSeeder.php
namespace Database\Seeders;

use Illuminate\Database\Seeder;
use App\Models\Author;
use App\Models\Book;
use App\Models\Genre;
use App\Models\Review;

class DatabaseSeeder extends Seeder
{
    public function run()
    {
        \App\Models\Author::factory(10)->create()->each(function ($author) {
            $books = \App\Models\Book::factory(3)->create(['author_id' =>
$author->id]);
            $books->each(function ($book) {
                $genres = \App\Models\Genre::factory(2)->create();
                $book->genres()->attach($genres);
```

```
                \App\Models\Review::factory(5)->create(['book_id' => $book-
>id]);
            });
        });
    }
}
```

This setup involves:

- Creating the necessary migration files to define the database schema.
- Defining the models and their relationships.
- Creating factories for generating fake data.
- Using a seeder to populate the database with realistic test data.

Run the migrations and seed the database to see the relationships in action:

```
php artisan migrate
php artisan db:seed
```