In Laravel's Eloquent ORM, a polymorphic relationship allows a model to belong to more than one other model on a single association. This is useful when you want to share a relationship across multiple models without duplicating the relationship logic.

Example: Polymorphic Relationship (One-to-Many)

Let's assume we have two models, `Post` and `Video`, that can both receive comments. Instead of creating two separate `comments` tables, we can use a polymorphic relationship to manage the comments for both models.

Step 1: Create Models

You will create three models:

- **Post**
- **Video**
- **Comment**

A `Comment` can belong to either a `Post` or a `Video`.

```
php artisan make:model Post -m
php artisan make:model Video -m
php artisan make:model Comment -m
```

Step 2: Define Migrations

`posts` table migration:

```
public function up() {
    Schema::create('posts', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->text('body');
        $table->timestamps();
    });
}
```

`videos` table migration:

```
public function up() {
    Schema::create('videos', function (Blueprint $table) {
        $table->id();
        $table->string('title');
        $table->string('url');
        $table->timestamps();
    });
}
```

`comments` **table migration** (Polymorphic relationship):

```
public function up() {
    Schema::create('comments', function (Blueprint $table) {
        $table->id();
        $table->text('body');
        $table->morphs('commentable'); // Creates 'commentable_id' and 'commentable_type'
        $table->timestamps();
    });
}
```

The `morphs('commentable')` function creates two columns:

- `commentable_id` (the ID of the model being commented on).
- `commentable_type` (the type of the model, either `Post` or `Video`).

Step 3: Define Models

Post Model (`Post.php`):

```
class Post extends Model {
    public function comments()
    {
        return $this->morphMany(Comment::class, 'commentable');
    }
}
```

Video Model (`Video.php`):

```
class Video extends Model {
    public function comments()
    {
        return $this->morphMany(Comment::class, 'commentable');
    }
}
```

Comment Model (`Comment.php`):

```
class Comment extends Model {
    public function commentable()
    {
        return $this->morphTo();
    }
}
```

Step 4: Usage Example

You can now associate comments with either posts or videos:

```
// Create a post
$post = Post::create(['title' => 'First Post', 'body' => 'This is the body of the post.']);

// Create a video
$video = Video::create(['title' => 'First Video', 'url' => 'http://example.com/video']);

// Add comments to the post
$post->comments()->create(['body' => 'Great post!']);
$post->comments()->create(['body' => 'Nice article!']);

// Add comments to the video
$video->comments()->create(['body' => 'Great video!']);
$video->comments()->create(['body' => 'I loved this content!']);

// Fetch comments from a post or video
$postComments = $post->comments;
$videoComments = $video->comments;
```

In this example, both `Post` and `Video` can have comments, but they share the `comments` table, allowing a flexible structure that can be extended to other models if needed.