ORM

Object Relational Mapping
with examples using Eloquent

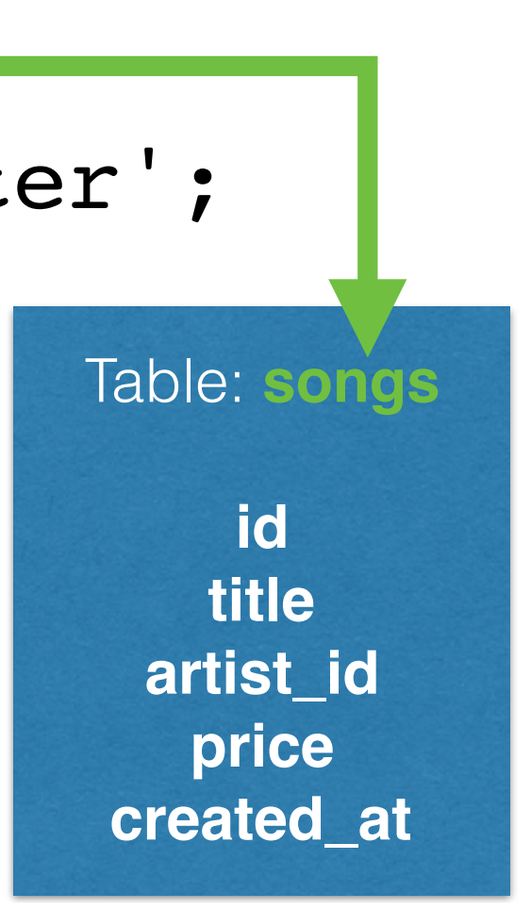
What is an ORM?

Object Relational Mapping

Examples

Create a song record in songs

```
$song = new Song();
$song->title = 'Sound of winter';
$song->artist_id = 25;
$song->price = 1.29;
$song->save();
```



ORM Advantages

Development speed - You don’t have to write low-level data access code.

Think about your database in an object oriented fashion

Don’t have to write SQL,
but you need to understand SQL!

Examples

Updating song record 5 in songs

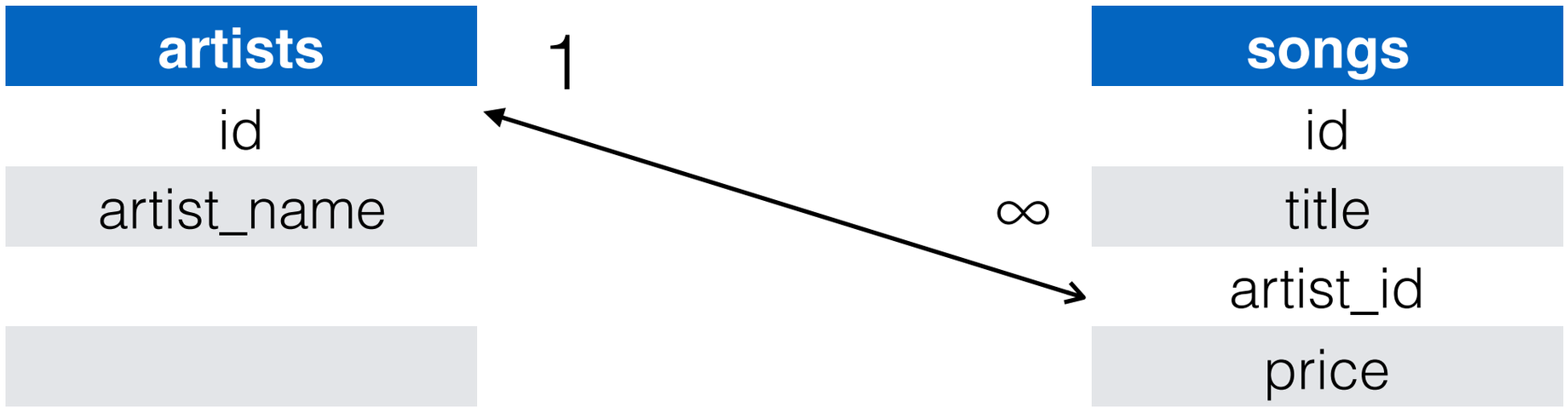
```
$song = Song::find(5);
$song->title = 'Sound of winter';
$song->price = 0.99;
$song->save();
```

Exercises

- Change database to “music-orm”
- Create an artist
- Create a song for that artist
- Update the song’s price
- Find the genre for that song

Relationships

One-to-Many Relationships



An artist has many songs

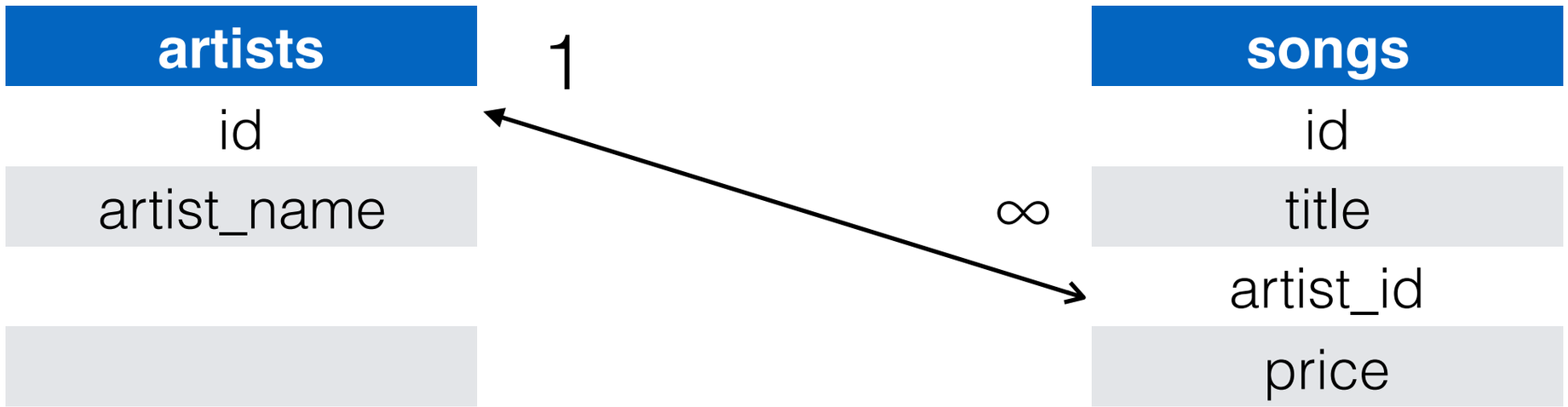
hasMany()

```php
class Artist extends Model {
  public function songs()
  {
    return $this->hasMany('App\Models\Song');
  }
}

$artist = Artist::find(5);
$songsByArtist = $artist->songs;
```

```
SELECT * FROM artists WHERE artists.id = 5 LIMIT 1
SELECT * FROM songs WHERE songs.artist_id = 5
```

One-to-Many Relationships



A song belongs to an artist

belongsTo()

```
$song = Song::all();
```

What about accessing the artist that each song belongs to?

```
class Song extends Model {
  public function artist()
  {
    return $this->belongsTo('App\Models\Artist');
  }
}
```

songs.artist_id = artists.id

```php
$songs = Song::take(5)->get();

foreach ($songs as $song) {
  // Artist model
   var_dump($song->artist);
}
```

Populating an object with data **on demand** (lazy loading)

N + 1 queries

6 queries

(1) SELECT * FROM songs LIMIT 5

(5) SELECT * FROM artist WHERE id = ?

Death by a Thousand Queries

- N + 1 problem might execute too many queries on a given page, which could be slow
- Lazy-loading is good if you have a small number of queries. Sometimes a few small queries are faster than one large, complex query

Eager Loading

- Load up models with related models. Also called hydration.

- Ex: Load up each Song model with the corresponding Artist model

Eager Loading belongsTo()

```
$songs = Song::with('artist', 'genre')
   ->take(5)
   ->get();

foreach ($songs as $song) {
   $song->artist; // Artist model
   $song->genre;  // Genre model
}
```

Eager Loading hasMany()

```
$artists = Artist::with('songs')
->take(5)
->get();
```

Exercises

- Create a hasMany relationship between Genre and Song

- Fetch all songs for genre 1

- Create a belongsTo relationship between Song and Genre

- Fetch the genre for song 9

- Fetch all genres. Eagerload all songs for each genre

Disadvantages

- Less flexibility depending on the ORM
- Performance i.e. N + 1 problem if you're not careful
- Developer productivity when learning an ORM

Popular ORMs

- Laravel: Eloquent
- Symfony: Doctrine, Propel
- Rails: ActiveRecord
- Java: Hibernate

Summary

- ORMs are great for rapid application development
- ORMs abstract away SQL so you can work with your database using objects
- Still need to understand SQL