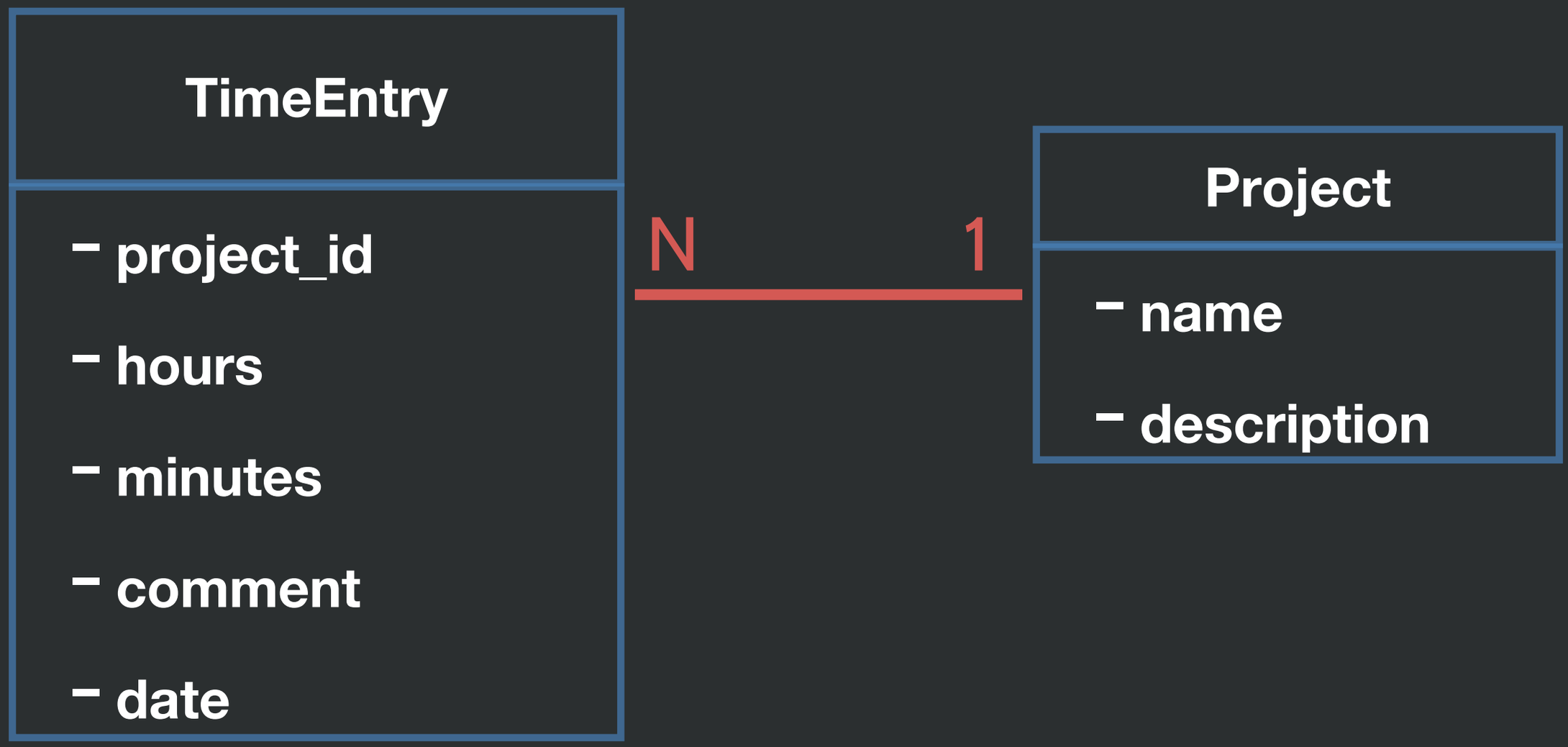
Introducing the model TimeEntry

Model: TimeEntry

Model: TimeEntry



Model: TimeEntry

```
$ rails g model TimeEntry
```

```
invoke  active_record
create    db/migrate/xxxxxxxxxxxxxx_create_time_entries.rb
create    app/models/time_entry.rb
```

Model: TimeEntry

app/models/time_entry.rb

```ruby
class TimeEntry < ActiveRecord::Base
end
```

Model: TimeEntry

db/migrate/xxxxxxxxxxxxxx_create_time_entries.rb

```
class CreateTimeEntries < ActiveRecord::Migration
  def change
    create_table :time_entries do |t|
      t.integer  :project_id
      t.integer  :hours
      t.integer  :minutes
      t.text     :comments
      t.datetime :date
      t.timestamps
    end
  end
end
```

Model: TimeEntry

Alternative way to define the foreign_key

```
class CreateTimeEntries < ActiveRecord::Migration
  def change
    create_table :time_entries do |t|
      t.references :project, index: true
      t.integer   :hours
      t.integer   :minutes
      t.text      :comments
      t.datetime  :date
      t.timestamps
    end
  end
end
```

Model: TimeEntry

$ rake db:migrate

```
==  CreateTimeEntries: migrating
================================================
-- create_table(:time_entries)
   -> 0.0162s
==  CreateTimeEntries: migrated (0.0163s)
================================================
```

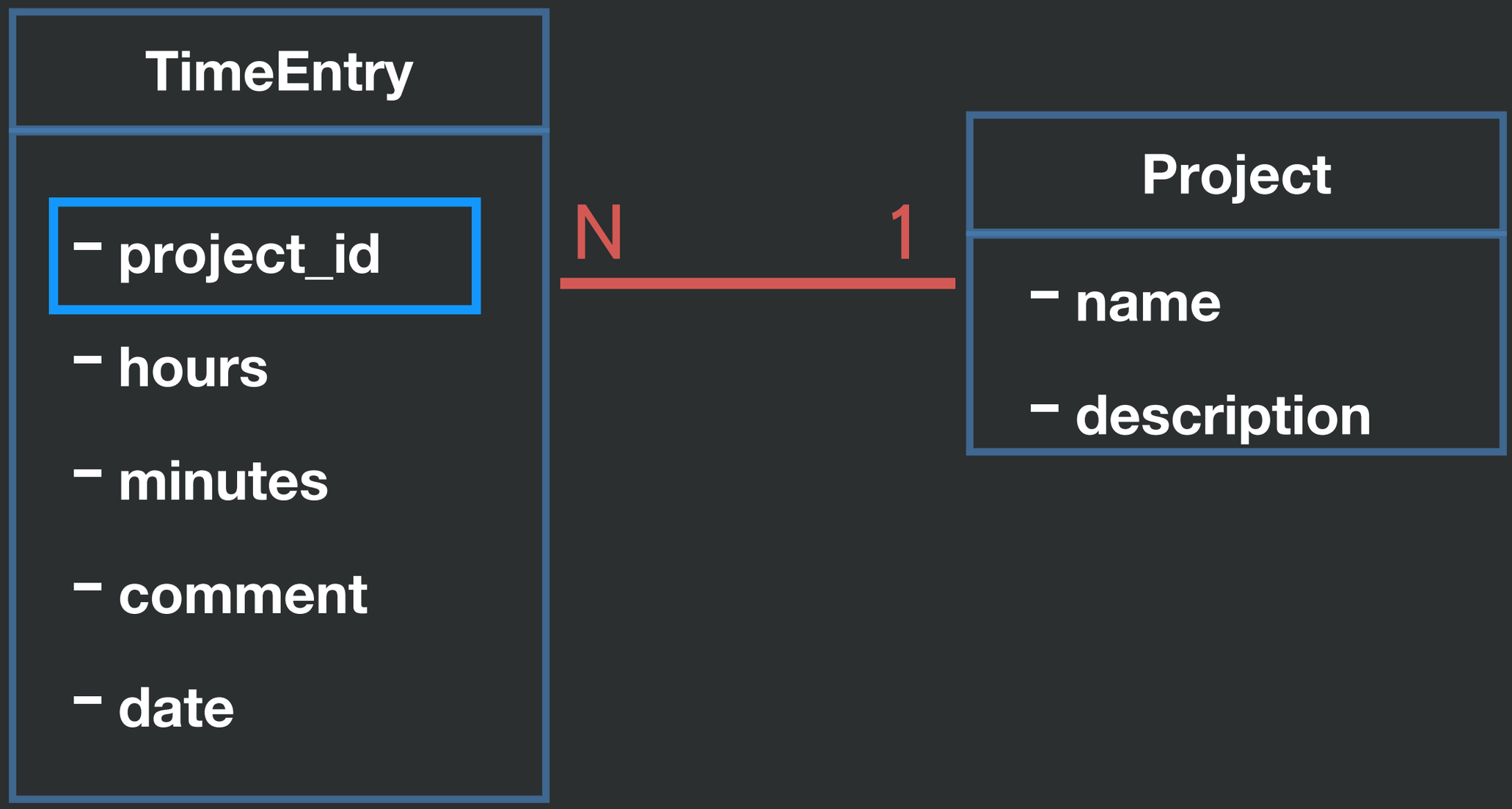
ActiveRecord Associations

ActiveRecord Associations

Provides an easy DSL to deal with relations between models.

Allow us to manage dependencies, such as cascade deletion.

ActiveRecord Associations



ActiveRecord Associations

```ruby
class Project < ActiveRecord::Base
  has_many :time_entries
end


class TimeEntry < ActiveRecord::Base
  belongs_to :project
end
```

ActiveRecord Associations

When defined they introduce lots of useful methods in our objects and classes.

There are several types:

- has_many
- belongs_to
- has_one
- has_and_belongs_to_many

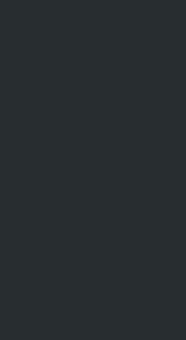
has_many / belongs_to

Showcase

has_many / belongs_to Showcase

```ruby
class Project < ActiveRecord::Base
  has_many :time_entries
end
```

```ruby
class TimeEntry < ActiveRecord::Base
  belongs_to :project
end
```



```
$ rails c

> project = Project.first
> project.time_entries
  => []

SELECT "time_entries".* FROM "time_entries"
WHERE "time_entries"."project_id" = $1
[["project_id", 1]]

  > project.id
  => 1
```

has_many / belongs_to Showcase

```
> project.time_entries.count
=> 0
```

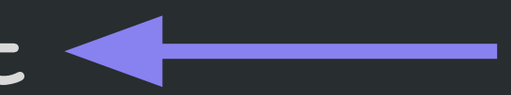
```
> project.time_entries.clear
# removes all time entries from project

> project.time_entries.empty?
=> true
```

has_many / belongs_to Showcase

```ruby
class Project < ActiveRecord::Base
  has_many :time_entries
end
```

```ruby
class TimeEntry < ActiveRecord::Base
  belongs_to :project
end
```



has_many / belongs_to Showcase

```
> entry = TimeEntry.new
> entry.hours = 0
> entry.minutes = 32
> entry.project = project
> entry.save
```

has_many / belongs_to Showcase

```
> entry.project
=> #<Project id: 1...>
```

has_many / belongs_to Showcase

```
> project.time_entries.create(hours: 0, minutes: 45)
```

```
> entry = project.time_entries.new
> entry.minutes = 32
> entry.hours = 1
> entry.save
```

has_many / belongs_to Showcase

```
> project.time_entries.count
> project.time_entries.empty?
> project.time_entries.create(hours: 0, minutes: 45)
```

What is `#time_entries` ?

What is #entries?

ActiveRecord::Associations::CollectionProxy

Automatically applies a scope on the entries so that all the entries belong to the parent project

Very useful for creation and fetching

has_many / belongs_to Showcase

```
> project.time_entries.where("minutes > 10").limit(3)
```

```
> TimeEntry
      .where("minutes > 10 AND project_id = ?", project.id)
       .limit(3)
```

Exercise

Update your db/seeds.rb with some time entries and associate them to a project.

Remember to commit your changes in git