中華電信學院
Ruby on Rails
後端之旅

簡煒航（大兜）@tonytonyjan

ORM

Object Relational Mapping
物件關聯對映

ORM 將關聯式資料庫以物件導向呈現

關聯式資料庫	資料表	欄位	單筆資料
物件導向	類別	屬性	物件

Q. 完成下表

Model / Class	Table / Schema
Post	ex. posts
LineItem	
Deer	
Mouse	
Person	

Rails 中的 ORM 實作為 ActiveRecord

$ rails dbconsole

$ rails dbconsole

```
hello $ rails dbconsole
SQLite version 3.8.5 2014-08-15 22:37:57
Enter ".help" for usage hints.
sqlite> CREATE TABLE posts(
   ...>    id INTEGER PRIMARY KEY,
   ...>    title TEXT,
   ...>    content TEXT
   ...> );
sqlite>
```

app/models/post.rb

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

$ rails console

```
Post.create title: "hello", content: "world"
```

$ rails dbconsole

```
sqlite> SELECT * FROM posts;
1|hello|world
```

$ rails console

```
x = Post.find 1

x.title = 'Goodbye'

x.save
```

$ rails dbconsole

```
sqlite> SELECT * FROM posts;
1|goodbye|world
```

SQL	ActiveRecord
SELECT * FROM posts;	Post.all
SELECT * FROM posts WHERE id = ‘1’	Post.find(1)
SELECT * FROM posts WHERE title = “hello”	Post.where(title: “hello”)

```ruby
# 新增
post = Post.create title: 'Hello', content: 'World'

# 更新
post = Post.find(1)
post.update title: 'New Title', content: 'New Content'

# 刪除
post = Post.find(1)
post.destroy
```

資料庫在哪？

資料庫位於
db/development.sqlite3
Rails 預設使用 SQLite

連接的資料庫可透過
config/database.yml
設定

config/database.yml

```yaml
development:
  adapter: sqlite3
  pool: 5
  timeout: 5000
  database: db/development.sqlite3
```

5xRuby 網站為例

```
production:
  adapter: postgresql
  encoding: unicode
  pool: 5
  database: fivexruby_production
  username: USERNAME
  password: PASSWORD
```

Q. 使用迴圈

產生 50 筆資料

ActiveRecord::Base

- 繼承於此的類別會得到豐富的擴充
- 以類別的複數小寫單字為資料表名（Post -> posts）
- 以資料表欄位為物件屬性
- 得到許多取代 SQL 的類別方法

Q. 若資料庫名稱不符合
Rails 命名慣例，
仍可與 Rails 整合嗎？

```ruby
class Article < ActiveRecord::Base
  self.table_name = 'posts'

end
```

資料庫遷移

$ rails g model user name email

db/migrate/xxx_create_users.rb

```ruby
class CreateUsers < ActiveRecord::Migration
  def change
    create_table :users do |t|
      t.string :name
      t.string :email

      t.timestamps
    end
  end
end
```

$ rake db:migrate

$ rails dbconsole

```
hello $ rails dbconsole
SQLite version 3.8.5 2014-08-15 22:37:57
Enter ".help" for usage hints.
sqlite> .tables
posts                schema_migrations  users
sqlite>
```

$ rake db:migrate:status

```
hello $ rake db:migrate:status

database: /private/tmp/hello/db/development.sqlite3

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20141021125710  Create users
```

$ rails g migration change_name

db/migrate/xxx_change_name.rb

```ruby
class ChangeName < ActiveRecord::Migration
  def change
    rename_column(:users, :name, :nickname)
  end
end
```

db/migrate/xxx_change_name.rb

```ruby
class ChangeName < ActiveRecord::Migration
  def up
    rename_column(:users, :name, :nickname)
  end

  def down
    rename_column(:users, :nickname, :name)
  end
end
```

```
hello $ rake db:migrate:status

database: /private/tmp/hello/db/development.sqlite3

  Status   Migration ID      Migration Name
--------------------------------------------------
    up     20141021125710  Create users
  down     20141021130539  Change name
```

```
hello $ rake db:migrate
== 20141021130539 ChangeName: migrating ======================================
-- rename_column(:users, :name, :nickname)
   -> 0.0055s
== 20141021130539 ChangeName: migrated (0.0056s) =============================

hello $ rake db:migrate:status

database: /private/tmp/hello/db/development.sqlite3

 Status   Migration ID    Migration Name
--------------------------------------------------
   up     20141021125710  Create users
   up     20141021130539  Change name

hello $
```

所有遷移方法都定義在
ActiveRecord::ConnectionAdapters::SchemaStatements

可逆的遷移都定義在ActiveRecord::Migration::CommandRecorder

Q. 為什麼要有 migration 的設計？

資料庫無法丟 git

ActiveRecord 操作

CRUD

Create

```ruby
# 使用 create
post = Post.create title: 'Hello', content: 'World'

# 使用 new 與 save
post = User.new # 不會存入資料庫
post.title = '標題'
post.content = '內容'
post.save # 存入資料庫
```

Read

```ruby
# 取得所有文章
posts = Post.all

# 取得第一篇文章
post = Post.first

# 尋找特定文章
post = Post.find_by(title: 'Hello')

# 尋找所有標題為 Hello 的文章，並按照新增時間排序
posts = Post.where(name: 'Hello').order('created_at DESC')
```

Update

```ruby
# 使用 save
post = Post.first
post.name = '哈蘿'
post.save

# 使用 update
post = Post.find_by(name: 'David')
post.update(name: 'Dave')

# 更新所有資料
Post.update_all title: '新標題'

# 局部更新所有資料
Post.where(title: '新標題').update_all title: '新新標題'
```

Destroy

```ruby
# 刪除單筆資料
post = Post.find_by(title: '標題')
post.destroy

# 刪除所有資料
Post.destroy_all
```

Q. 局部刪除所有資料？

徒手製造 Scaffold

$ rails g scaffold NAME ATTRIBUTE...

config/routes.rb

```ruby
Rails.application.routes.draw do
  resources :posts
end
```

$ rake routes

```
hello $ rake routes
    Prefix Verb   URI Pattern                  Controller#Action
     posts GET    /posts(.:format)             posts#index
           POST   /posts(.:format)             posts#create
  new_post GET    /posts/new(.:format)         posts#new
 edit_post GET    /posts/:id/edit(.:format)    posts#edit
      post GET    /posts/:id(.:format)         posts#show
           PATCH  /posts/:id(.:format)         posts#update
           PUT    /posts/:id(.:format)         posts#update
           DELETE /posts/:id(.:format)         posts#destroy
```

Read

app/controllers/posts_controller.rb

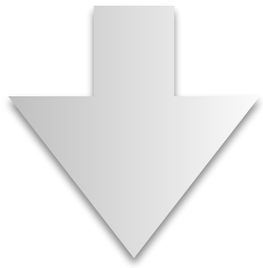
```ruby
class PostsController < ApplicationController
  def index
    @posts = Post.all
  end

  def show
    @post = Post.find(params[:id])
  end
end
```

Q. 完成 view 的部分

Create

app/views/posts/new.html.erb

```erb
<!-- app/views/posts/new.html.erb -->
<%= form_tag posts_path do %>
  <%= text_field_tag :title %>
  <%= text_field_tag :content %>
  <%= submit_tag %>
<% end %>
```



```html
<input id="title" name="title" type="text" />
<input id="content" name="content" type="text" />
<input name="commit" type="submit" value="Save changes" />
```

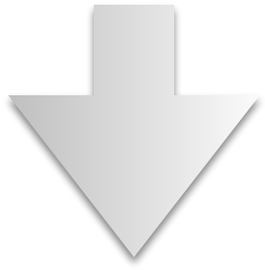
app/controllers/posts_controller.rb

```ruby
def create
  @post = Post.new({
    title: params[:title],
    content: params[:content]
  })
  if @post.save
    redirect_to post_path(@post.id)
  else
    render :new
  end
end
```

可以更好

app/views/posts/new.html.erb

```erb
<!-- app/views/posts/new.html.erb -->
<%= form_tag posts_path do %>
  <%= text_field_tag 'post[title]' %>
  <%= text_field_tag 'post[content]' %>
  <%= submit_tag %>
<% end %>
```



```html
<input id="post_title" name="post[title]" type="text" />
<input id="post_content" name="post[content]" type="text" />
<input name="commit" type="submit" value="Save changes" />
```

params[:post].is_a? Hash

```
Started POST "/posts" for 127.0.0.1 at 2014-10-22 17:12:59 +0800
Processing by PostsController#create as HTML
  Parameters: {"utf8"=>"✓", "authenticity_token"=>"ii/XB2sbTHe9SNNH1mTPFZ2jLKl0Aw5h
SiFxBgTBO58=", "post"=>{"title"=>"Hello", "content"=>"World"}, "commit"=>"Save chan
ges"}
   (0.1ms)  begin transaction
   (0.1ms)  rollback transaction
  Rendered posts/new.html.erb within layouts/application (1.0ms)
Completed 200 OK in 31ms (Views: 24.5ms | ActiveRecord: 0.6ms)
```

app/controllers/posts_controller.rb

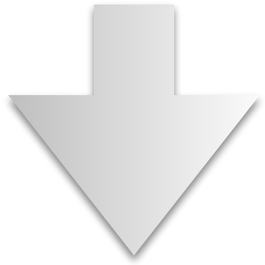
```ruby
def create
  @post = Post.new params[:post].to_h
  if @post.save
    redirect_to post_path(@post.id)
  else
    render :new
  end
end
```

Q. 可能有什麼危險？

```
def create
  @post = Post.new params[:post].to_h
  if @post.save
    redirect_to post_path(@post.id)
  else
    render :new
  end
end
```

並不是所有 column 都可讓使用者存取

```
<input id="user_email" name="user[email]" type="text" />
<input id="user_name" name="user[name]" type="text" />
<input id="user_is_admin" name="user[is_admin]" type="text" />
<input name="commit" type="submit" value="Save changes" />
```



```
Started POST "/posts" for 127.0.0.1 at 2014-10-22 17:24:08 +0800
Processing by PostsController#create as HTML
  Parameters: {"utf8"=>"✓", "authenticity_token"=>"ii/XB2sbTHe9SNNH1mTPFZ2jLKl0Aw5hSiFxBgTBO58=", "
user"=>{"email"=>"tonytonyjan@gmail.com", "name"=>"tonytonyjan", "is_admin"=>"true"}, "commit"=>"Sa
ve changes"}
   (0.0ms)  begin transaction
   (0.0ms)  rollback transaction
  Rendered posts/new.html.erb within layouts/application (0.4ms)
Completed 200 OK in 32ms (Views: 26.2ms | ActiveRecord: 0.5ms)
```

Strong Parameter

```ruby
def create
  @post = Post.new params.require(:post).permit(:title, :content)
  if @post.save
    redirect_to post_path(@post.id)
  else
    render :new
  end
end
```

params 必須含有 post，
且 post 裡面只有 title 和 content 可以通過，
其餘會被過濾掉

form_for

```erb
<!-- app/views/posts/new.html.erb -->
<%= form_for @post do |f| %>
  <%= f.text_field :title %>
  <%= f.text_field :content %>
  <%= f.submit %>
<% end %>
```

Update

app/views/posts/edit.html.erb

```
<!-- app/views/posts/edit.html.erb -->
<%= form_for @post do |f| %>
  <%= f.text_field :title %>
  <%= f.text_field :content %>
  <%= f.submit %>
<% end %>
```

app/controllers/posts_controller.rb

```ruby
# GET /posts/:id/edit
def edit
  @post = Post.find(params[:id])
end

# PUT /posts/:id
def update
  @post = Post.find(params[:id])
  if @post.update params.require(:post).permit(:title, :content)
    redirect_to @post
  else
    render :edit
  end
end
```

重複的部分

```ruby
# GET /posts/:id/edit
def edit
  @post = Post.find(params[:id])
end

# PUT /posts/:id
def update
  @post = Post.find(params[:id])
  if @post.update params.require(:post).permit(:title, :content)
    redirect_to @post
  else
    render :edit
  end
end
```

app/controllers/posts_controller.rb

```ruby
private
def post_params
  params.require(:post).permit(:title, :content)
end

def set_post
  @post = Post.find(params[:id])
end
```

定義在 private 底下，以區別 action

before_action

```ruby
before_actoin :set_post, only: %i[show edit update]
def show
end

def edit
end

def update
  if @post.update post_params
    redirect_to @post
  else
    render :edit
  end
end
```

在 action 執行之前先執行指定的方法

Destroy

app/views/posts/show.html.erb

```
<%= link_to 'delete', @post, method: :delete %>
```

Q. 製作 controller 的部分

Validation

資料驗證

app/models/post.rb

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
  validates :title, presence: true
end
```

$ rails console

```
post = Post.new

post.save # => false

post.title = 'Hello'

post.save # => true
```

各種驗證

```ruby
# 一定為真（用在 boolean）
validates :terms, acceptance: true
# 正規表達式
validates :email, format: { with: /\A([^@\s]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i }
# 範圍之外
validates :username, exclusion: { in: %w(admin superuser) }
# 範圍之內
validates :age, inclusion: { in: 0..9 }
# 長度
validates :first_name, length: { maximum: 30 }
# 必須是數字
validates :age, numericality: true
# 必填
validates :username, presence: true
# 不能與資料庫中的重複
validates :username, uniqueness: true
```

Q. 文章標題必須以數字開頭

Callback

回呼

使用方式（1/2）

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
  validates :title, presence: true

  before_create do
    self.content = title if content.blank?
  end
end
```

使用方式（2/2）

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
  validates :title, presence: true
  before_create :generate_content_from_title

  protected
    def generate_content_from_title
      self.content = title if content.blank?
    end
end
```

create

- before_validation
- after_validation
- before_save
- around_save
- before_create
- around_create
- after_create
- after_save

update

- before_validation
- after_validation
- before_save
- around_save
- before_update
- around_update
- after_update
- after_save

destroy

- before_destroy
- around_destroy
- after_destroy

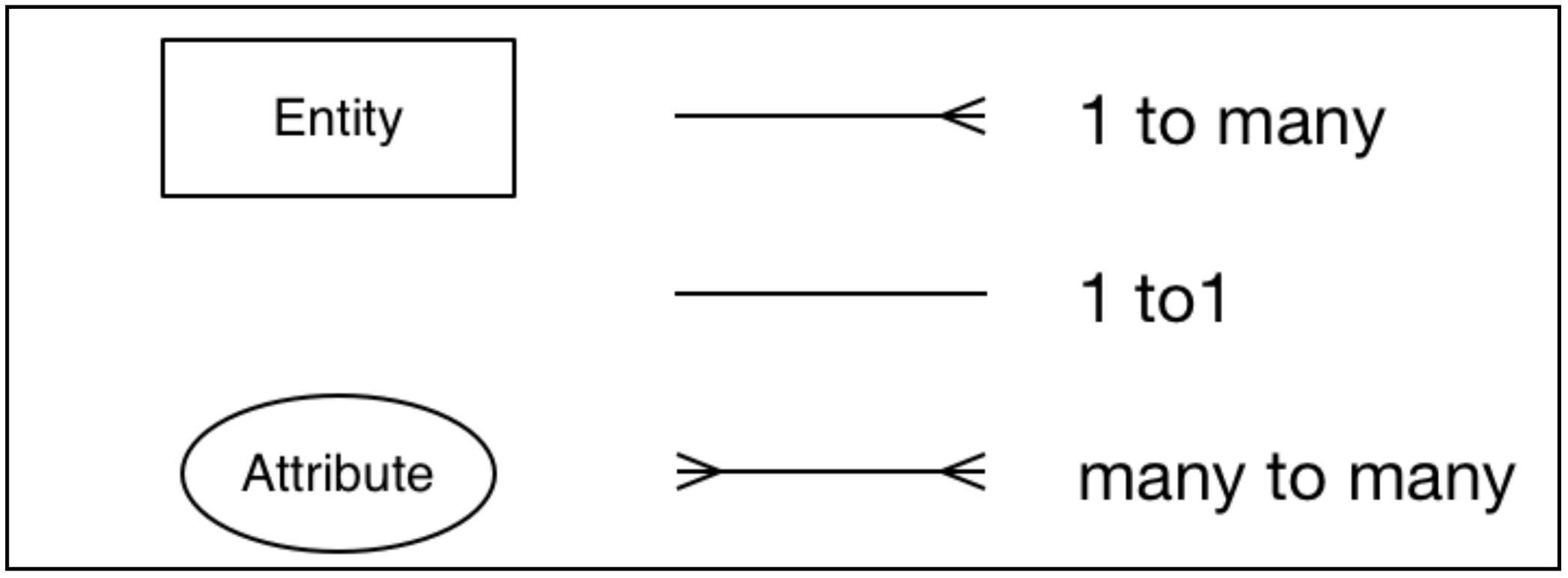
Q. 在文章儲存前將所有文字變大寫

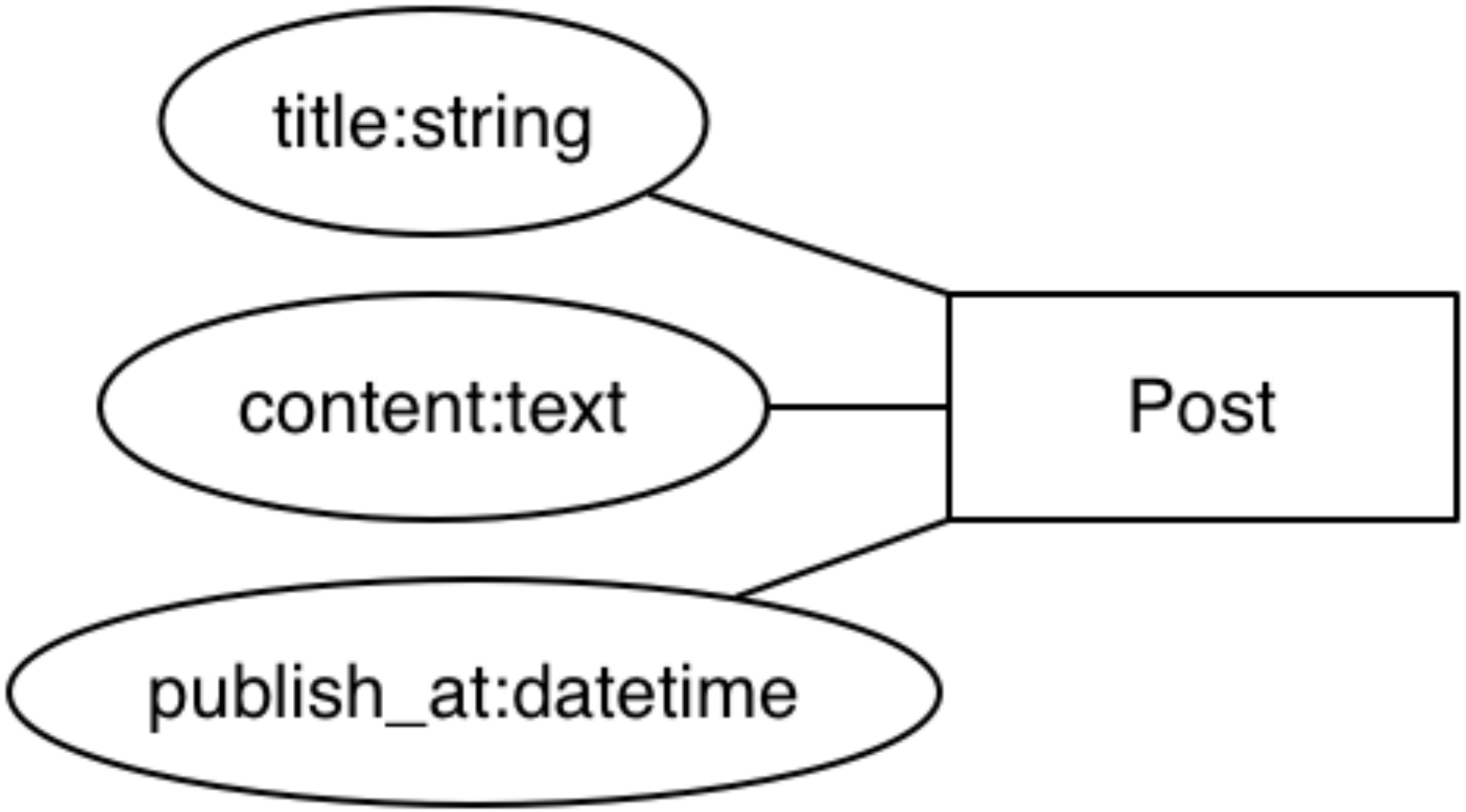
E-R Model

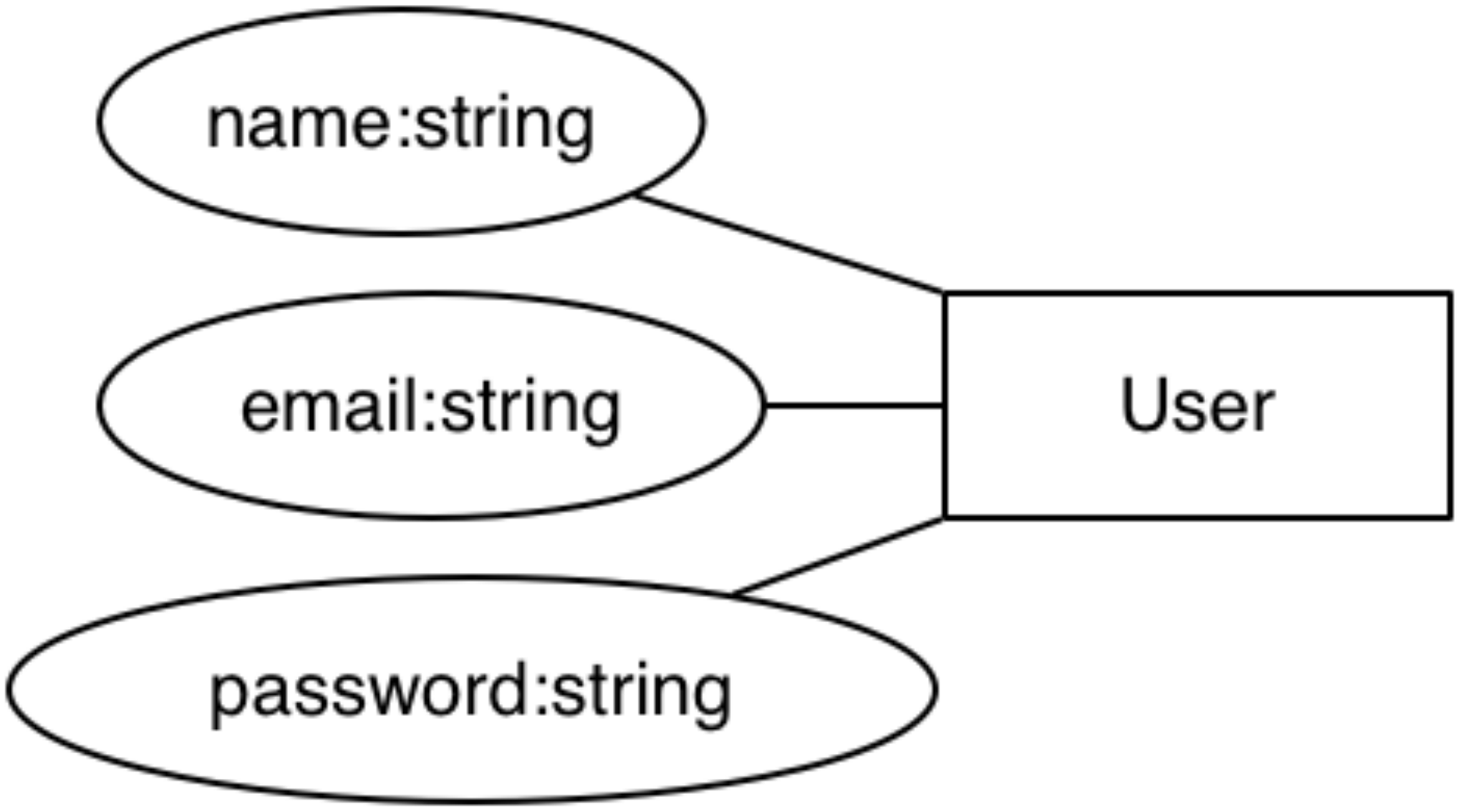
Entities-Relational Model

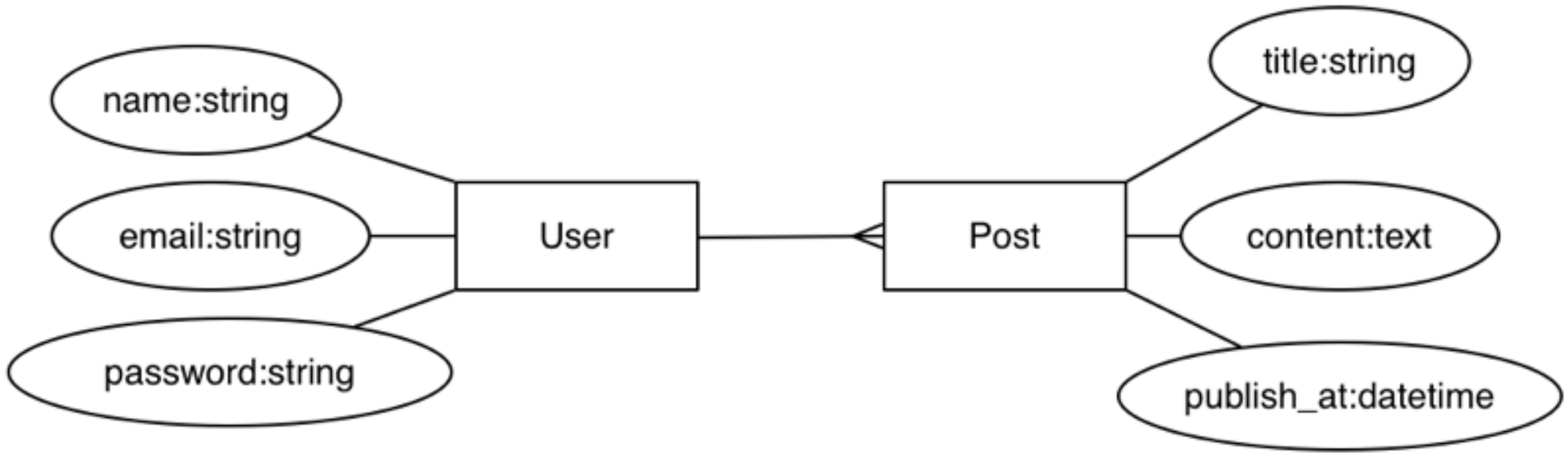
個體-關係模式

符號介紹

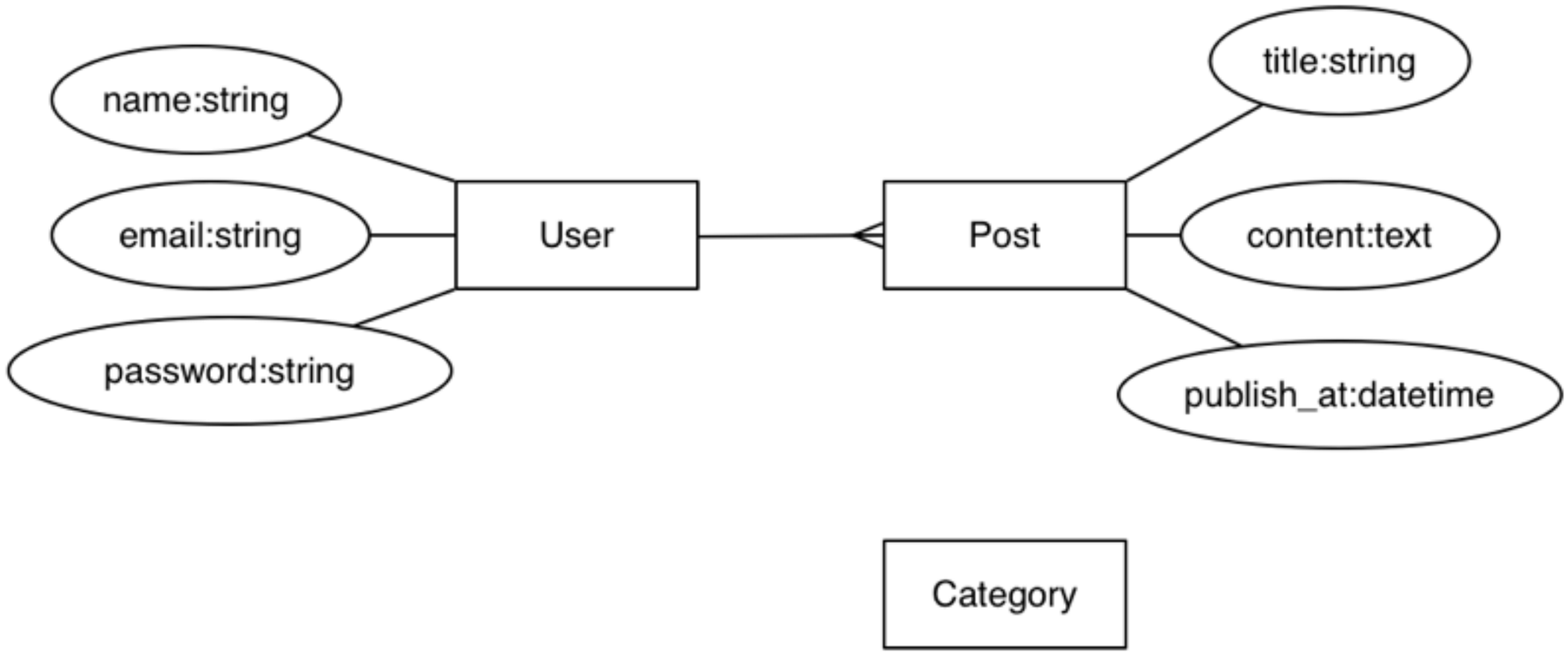


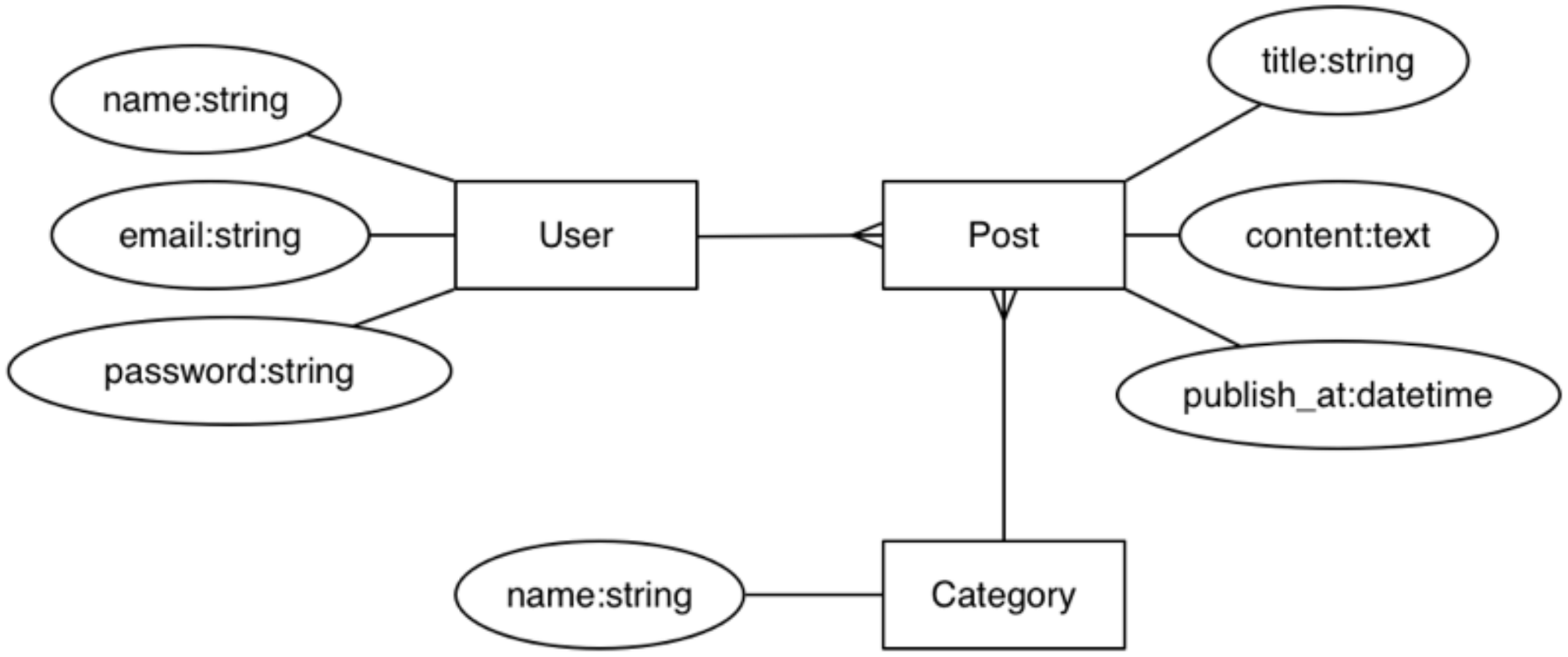

title:string
content:text
publish_at:datetime
Post


name:string
email:string
password:string
User

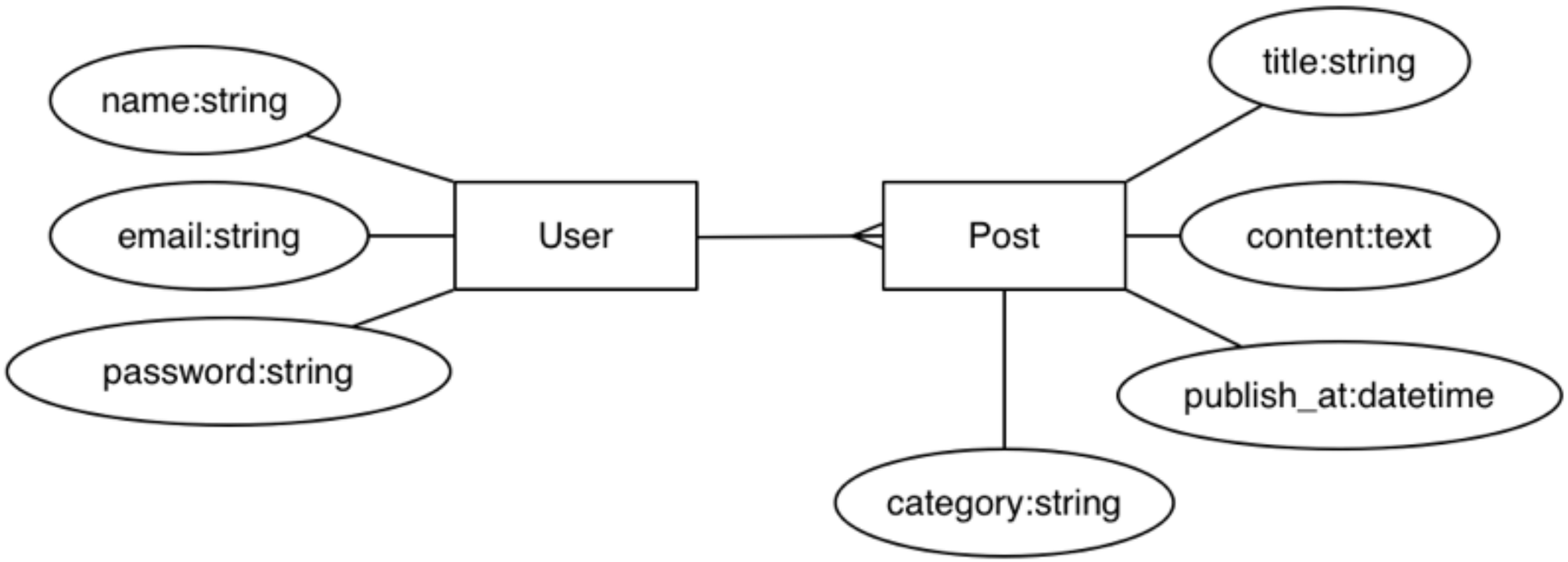

name:string
email:string
password:string
User
title:string
content:text
publish_at:datetime
Post

Q. 加上分類

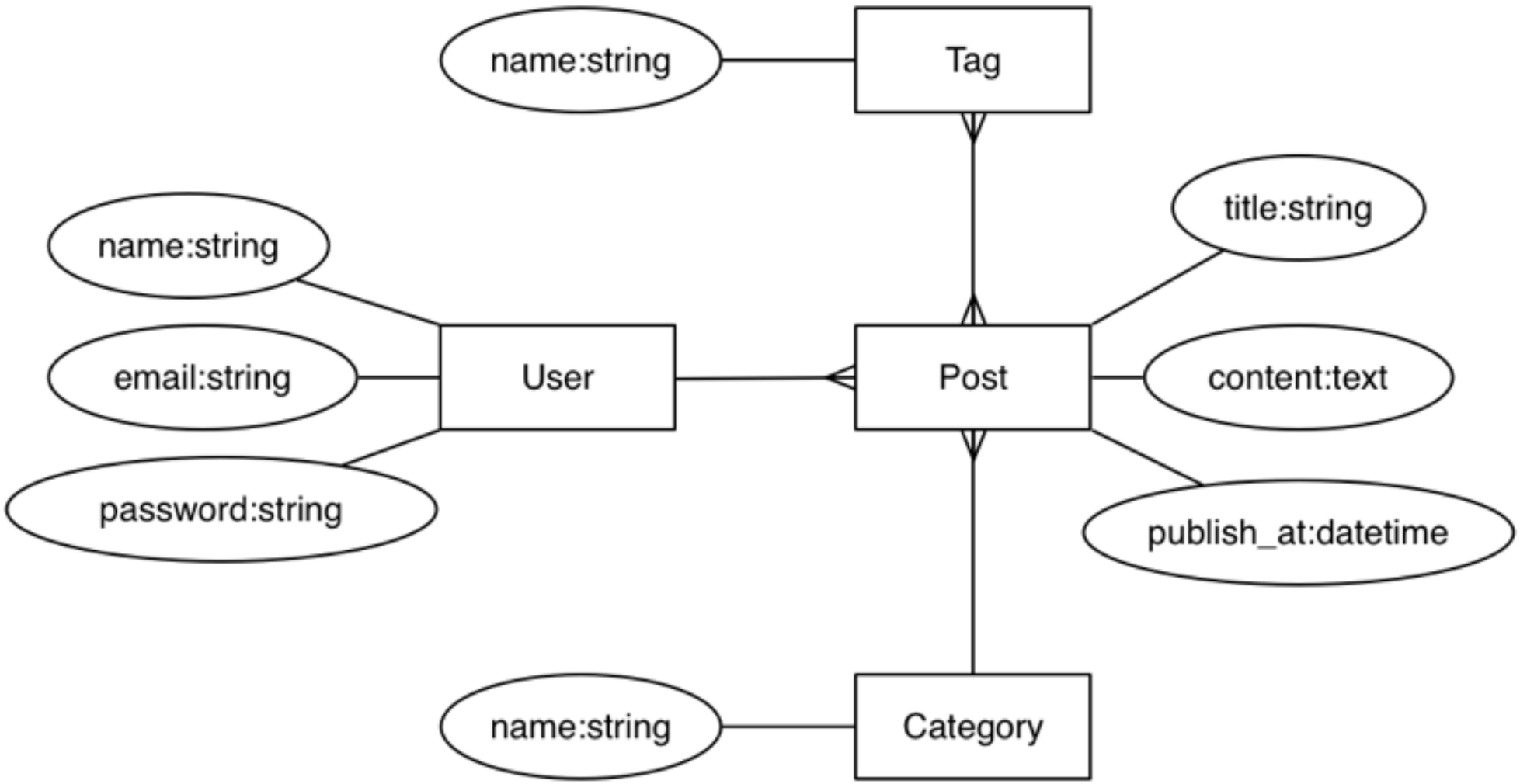

name:string
email:string
password:string
User
title:string
Post
content:text
publish_at:datetime
Category


name:string
email:string
password:string
User
title:string
content:text
publish_at:datetime
Post
name:string
Category

Q. 什麼情況下會這樣設計



加上標籤


name:string
Tag
name:string
title:string
email:string
User
Post
content:text
password:string
publish_at:datetime
name:string
Category

屬性須是基本資料

- string - 短字串（255）
- text - 長字串
- date - 日期
- time - 時間
- datetime - 日期時間
- boolean - 真假值
- integer - 整數
- float - 浮點數
- decimal - 高精浮點數

ERM 轉 RMDBS

posts 資料表

id	title	content
1	哈囉	世界
2	天氣不錯	適合學 Rails
3	天氣不好	也是合學 Rails
…	…	…

users 資料表

id	name	email
1	大兜	tony@5xruby.tw
2	大雄	nobita@5xruby.tw
3	大大	dada@5xruby.tw
…	…	…

Q. 如何建立一對多關聯？

posts 資料表

id	title	content	user_id
1	哈囉	世界	1
2	天氣不錯	適合學 Rails	1
3	天氣不好	也適合學 Rails	2
…	…	…	

user_id 又稱 forign_key，用於參照其他資料表

foreign key 命名慣例

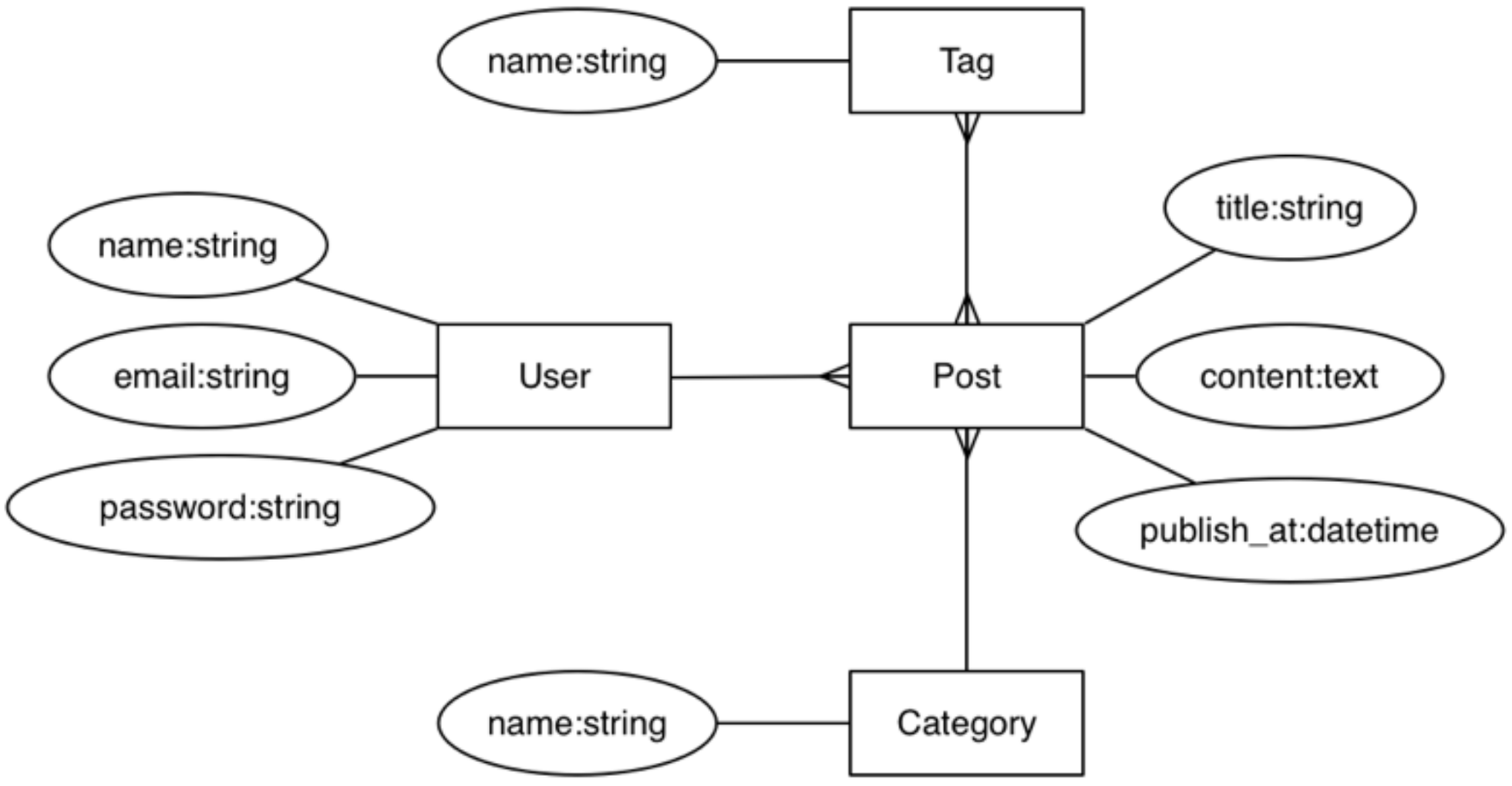
“單數_id”

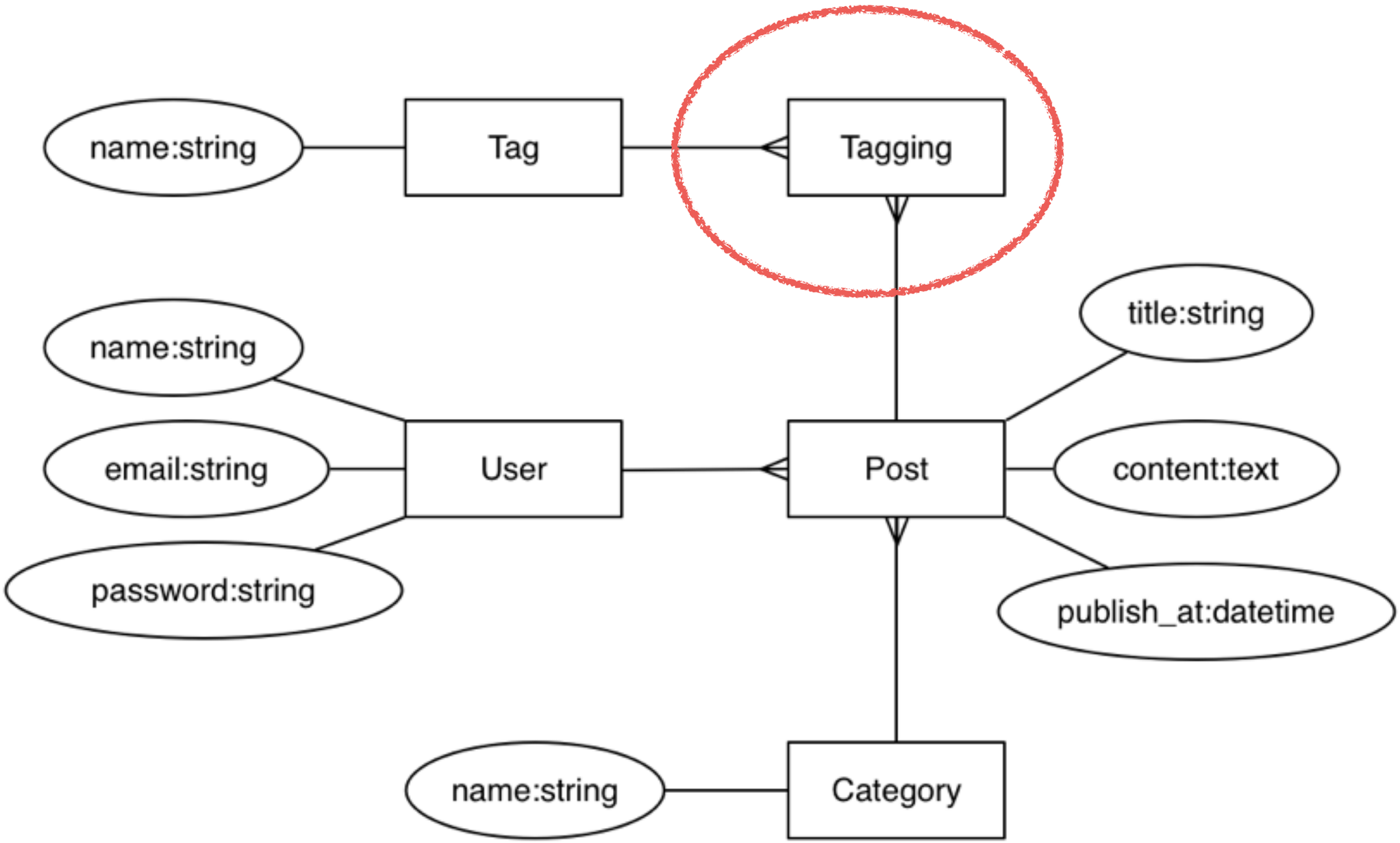
Tag?

Tag 資料表

id	name
1	心情
2	工作
3	有感
…	…

Q. 如何建立多對多關聯？


name:string
Tag
name:string
title:string
email:string
User
Post
content:text
password:string
publish_at:datetime
name:string
Category


name:string
Tag
Tagging
name:string
title:string
email:string
User
Post
content:text
password:string
publish_at:datetime
name:string
Category

Tagging 資料表

tag_id	post_id
1	2
1	3
2	4
…	…

Association
資料關聯

```
$ rails new blog
$ cd blog
$ rails g scaffold User name email
$ rails g scaffold Post title content:text user:references
$ rake db:migrate
```

db/migrate/xxx_create_posts.rb

```ruby
class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :title
      t.string :content
      t.references :user, index: true
      t.timestamps
    end
  end
end
```

相等的寫法

```ruby
class CreatePosts < ActiveRecord::Migration
  def change
    create_table :posts do |t|
      t.string :title
      t.string :content
      t.integer :user_id
      t.index :user_id
      t.timestamps
    end
  end
end
```

app/models/post.rb

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
  belongs_to :user
end
```

app/models/user.rb

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  has_many :posts
end
```

post.user	user.posts
回傳 User 物件	回傳 Post 集合
SELECT * FROM users WHERE id = ?;	SELECT * FROM posts WHERE user_id = ?;

建立關聯（1/3）

```
user = User.create name: 'Tony', email: 'tony@5xruby.tw'
post = Post.create title: 'Hello', content: 'World'
post.update user_id: user.id
```

建立關聯（2/3）

```ruby
user = User.create name: 'Tony', email: 'tony@5xruby.tw'
post = Post.create title: 'Hello', content: 'World'
post.update user: user
```

建立關聯（3/3）

```ruby
user = User.create name: 'Tony', email: 'tony@5xruby.tw'
user.posts.create title: 'Hello', content: 'World'
```

Q. 在 /posts 顯示作者在 /users/:id 顯示文章

app/views/posts/_form.html.erb

```erb
<div class="field">
  <%= f.label :user_id %><br>
  <%= f.select :user_id, [['Tony', 1], ['Jason', 2]] %>
</div>
```

Q. 把所有 User 放入下拉選單

多對多

$ rails g model Tag name
$ rails g migration CreateTagJoinTable tags posts
$ rake db:migrate

資料表：posts_tags

欄位：post_id, tags_id

app/models/post.rb

```ruby
# app/models/post.rb
class Post < ActiveRecord::Base
  belongs_to :user
  has_and_belongs_to_many :tags
end
```

app/models/tag.rb

```ruby
class Tag < ActiveRecord::Base
  has_and_belongs_to_many :posts
end
```

Q. 在 rails console 手動幫文章建立標籤，並在 /posts/:id 顯示標籤

會員驗證

$ rails g migration add_password_to_users password_digest:string

這是一行

Gemfile

gem 'bcrypt', '~> 3.1.7'

bundle install 後重開 server

app/models/user.rb

```ruby
# app/models/user.rb
class User < ActiveRecord::Base
  has_many :posts
  has_secure_password
end
```

基本用法

```
user = User.new(name: 'tony', password: '', password_confirmation: 'nomatch')
user.save # => false, password required
user.password = '12345678'
user.save # => false, confirmation doesn't match
user.password_confirmation = '12345678'
user.save # => true
user.authenticate('notright') # => false
user.authenticate('12345678') # => user
User.find_by(name: 'tony').try(:authenticate, 'notright') # => false
User.find_by(name: 'tony').try(:authenticate, '12345678') # => user
```

app/views/users/_form.html.erb

```
<div class="field">
  <%= f.label :password %><br>
  <%= f.password_field :password %>
</div>
<div class="field">
  <%= f.label :password_confirmation %><br>
  <%= f.password_field :password_confirmation %>
</div>
```

app/controllers/users_controller.rb

```
def user_params
  params.require(:user).permit(
    :name, :email, :password, :password_confirmation
  )
end
```

config/routes.rb

```ruby
Rails.application.routes.draw do
  get    'sign_in'  => 'sessions#new'
  post   'sign_in'  => 'sessions#create'
  delete 'sign_out' => 'sessions#destroy'
  resources :posts
  resources :users
end
```

$ rails g controller sessions new

app/views/sessions/new.html.erb

```erb
<h1>Sign in</h1>
<%= form_tag sign_in_path do %>
  <input type="text" name="name">
  <input type="password" name="password">
  <input type="submit">
<% end %>
```

app/controllers/sessions_controller.rb

```ruby
class SessionsController < ApplicationController
  def create
    if user = User.find_by(name: params[:name]).try(:authenticate, params[:password])
      session[:user_id] = user.id
      redirect_to root_path
    else
      render :new
    end
  end

  def destroy
    session[:user_id] = nil
    redirect_to root_path
  end
end
```

app/views/layouts/application.html.erb

```erb
<% if user_signed_in? %>
  <%= link_to 'Sign Out', sign_out_path, method: :delete %>
<% else %>
  <%= link_to 'Sign In', sign_in_path %>
<% end %>

<%= yield %>
```

app/controllers/application_controller.rb

```ruby
class ApplicationController < ActionController::Base
  protect_from_forgery with: :exception
  helper_method :current_user, :user_signed_in?
  protected
  def current_user
    User.find_by(id: session[:user_id])
  end

  def user_signed_in?
    !current_user.nil?
  end
end
```

限制發文

Q. 實作 auth_user 部分

```
class PostsController < ApplicationController
  before_action :auth_user, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_post, only: [:show, :edit, :update, :destroy]
```

Q. 限制登入者權限

```ruby
class PostsController < ApplicationController
  before_action :auth_user, only: [:new, :create, :edit, :update, :destroy]
  before_action :set_post, only: [:show, :edit, :update, :destroy]
  before_action :check_owner, only: [:edit, :update, :destroy]
```

只有文章作者可以更新、刪除文章

```ruby
def auth_user
  redirect_to root_path unless user_signed_in?
end

def check_owner
  redirect_to root_path if @post.user != current_user
end
```

資源參考

ruby.tw

Ruby Taiwan

- **Twitter 推特**
- **Blog 部落格**
- **Jobs 工作**
- **GitHub 開源專案**
- **Ruby 中文官網**
- **RubyConf Taiwan 大會**
- **Facebook 專頁**
- **Facebook 社團**

Groups

- **Rails Taiwan Meetup** by xdite, Jerry Lee
- **Rails Taiwan**
- **Rails Girls Taiwan**
- **Ruby on Rails 讀書會** by marsz, Rich Ke
- **Ruby on Rails 新手村**
- **RailsFun.tw**
- **Google+ 社群** by 簡煒航
- **PTT BBS #Ruby 版** by breakanyrule
- **Ruby&Rails Fun!!** by lovingzatz

英文資源

- Rails Guide - http://guides.rubyonrails.org/
- Rails API - http://api.rubyonrails.org/
- Rails Tutorial - https://www.railstutorial.org/
- RailsCast - http://railscasts.com/
- Rails for Zombies - http://railsforzombies.org/

指令整理

- rails dbconsole
- rails console
- rails generate scaffold NAME ATTRIBUTE…
- rake db:migrate
- rake db:migrate:status
- rake db:rollback