Web Application Architectures

Module 3: Database Interactions

Lecture 4: The Blog App – Iteration 2 (Associations)

THE UNIVERSITY *of* NEW MEXICO

Associations in Rails

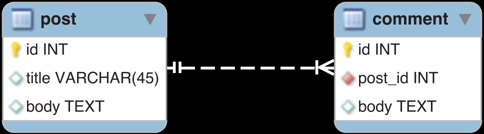
Because we used the scaffold generator for the posts and comments in our blog application, the Active Record design pattern is “pre-wired.” This means:

- A SQLite database able to store posts and comments was created when we ran the migrations.
- A connection to this database was established.
- The ORM for `Post` and `Comment` objects was set up — the “M” in MVC.

However, there’s one thing missing — we need to ensure that any comments entered for a particular post are permanently linked to that post.

Associations in Rails

- We did make the connection between posts and comments in the database — recall that a `post_id` can be stored with each comment.
- To make our models in Rails fully functional we need to add associations — each post needs to know the list of comments associated with it, and each comment needs to know which post it belongs to.
- There's a many-to-one relationship between comments and posts — a post has many comments, and a comment belongs to a post:



Associations in Rails

- The `ActiveRecord` module contains a set of class methods for tying objects together through foreign keys.
- To enable these, you must declare associations within your models using the following:

Relationship	Model with no foreign key	Model with foreign key
one-to-one	`has_one`	`belongs_to`
many-to-one	`has_many`	`belongs_to`
many-to-many	`has_and_belongs_to_many`	⋆

⋆ The foreign keys for each model are stored in a join table.