Week 5 Lecture 13

Applied

Helpful Resources

- http://en.wikipedia.org/wiki/Join_(SQL)
- http://www.codinghorror.com/blog/2007/10/a-visual-explanation-of-sql-joins.html
- http://magicscalingsprinkles.wordpress.com/2010/01/28/why-i-wrote-arel/
- http://api.rubyonrails.org/classes/ActiveRecord/Associations/ClassMethods.html

What's in this lecture?

- Advanced DataBase Structure
- Understanding Joins
- Helpful Tips

Advanced Database Structure

What we know so far

- Tables map to individual data models
- Records correspond to instances of models
- Separation of data by tables

What we need

- A way to define relationships between data
- Leverage associations between data

Related Data

- Applications have related data that:
  - defines a relationship
    Posts **have** comments
    Projects **have** owners
  - extends a data model
    Users **have** ContactInfo
    Pictures **have** Descriptions

Associations

- Records have known (or hidden!) associations that
  - Allow for advanced filtering
    Users **who have** more than 10 photos
    Applications that **are missing** resumes
  - Retrieve sets of like data
    Logins **and** actions **for a given** date
    Users **who voted** for an article

Defining Relationships

- Relationships are defined by primary keys and foreign keys
  - Primary Keys are unique to a table
  - Foreign keys are the primary keys of a different table

Simple Posts Example

Posts

PostID	Title	UserID
23	“Food!”	13
...	...	...

Users

UserID	Name
13	“Kip”

Simple Post Example

- In the Posts table:
  - PostID is the primary key
  - User ID is the foreign key
- In the Users table:
  - User ID is the primary key
  - there are no foreign keys

Note:

- Primary keys are a special data type
- No requirement to distinguish foreign keys from other data types
- ‘foreign’ and ‘primary’ are context based

Advanced Selections

- Posts by Author:
  SELECT title FROM posts p WHERE p.user_id = 13
- Author of a Post:
  SELECT name FROM users u WHERE u.user_id = 13

The Join

Use Case

- What if we wanted to list all users and their posts?

```
SELECT Users.name, Posts.title
FROM Users
INNER JOIN Posts
ON users.user_id=posts.user_id;
```

What's happening here?

- **SELECT Users.name, Posts.title**
  - Columns we want returned
- **FROM Users**
  - Our 'left' table
- **INNER JOIN Posts**
  - Our 'right' table
- **ON users.user_id=posts.user_id;**
  - Conditions on which the data should be matched

Thinking with Sets:

- (intentionally left blank)

Helpful Tips

- Seeing the generated SQL behind ARel:
  - Model.where(:condition=>value).to_sql
- The ActiveRecord docs are your friends
- Creating a sample db is fast and easy -- try an idea out to help think out a problem

Exercises

- Extend your two SQLite databases from last time to include primary/foreign key relationships
- For your databases list at least 2 situations where you'd use a JOIN