References

Agile Web Development with Rails, Thomas & Hanson, The Pragmatic Bookshelf, 2005

Active Record Documentation, http://ar.rubyonrails.com/
Databases

MySQL
http://www.mysql.com/
Open source
Later versions support Transactions

create database lectureExamples;

CREATE TABLE faculty (  
  name CHAR(20) NOT NULL,  
  faculty_id INTEGER AUTO_INCREMENT PRIMARY KEY
);

PostgreSQL
http://www.postgresql.org/
Open source
Now runs on Windows

create database lectureExamples;

CREATE TABLE faculty (  
  name CHAR(20) NOT NULL,  
  faculty_id SERIAL PRIMARY KEY
);
Converting Types

<table>
<thead>
<tr>
<th>SQL Type</th>
<th>Ruby Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>int, integer</td>
<td>Fixnum</td>
</tr>
<tr>
<td>decimal, numeric</td>
<td>Float</td>
</tr>
<tr>
<td>interval, date</td>
<td>Date</td>
</tr>
<tr>
<td>clob, blob, text</td>
<td>String</td>
</tr>
<tr>
<td>float, double</td>
<td>Float</td>
</tr>
<tr>
<td>char, varchar, string</td>
<td>String</td>
</tr>
<tr>
<td>datetime, time</td>
<td>Time</td>
</tr>
<tr>
<td>boolean</td>
<td>special handling needed</td>
</tr>
</tbody>
</table>
Normalization

First Normal Form
- Keep attributes single valued
- Put collections in separate table

<table>
<thead>
<tr>
<th>Books</th>
<th>Books</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>Title</td>
<td>Author1</td>
</tr>
<tr>
<td>1</td>
<td>Foo</td>
<td>you</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Join Table

<table>
<thead>
<tr>
<th>book_id</th>
<th>author_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Normalization

Second Normal Form
All non-key attributes must be fully dependent on the entire primary key
Don't repeat data

<table>
<thead>
<tr>
<th>Books</th>
<th>Books</th>
<th>Publishers</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>title</td>
<td>publisher</td>
</tr>
<tr>
<td>1</td>
<td>Foo</td>
<td>you</td>
</tr>
<tr>
<td>2</td>
<td>Bar</td>
<td>you</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>title</td>
<td>publisher_id</td>
</tr>
<tr>
<td>1</td>
<td>Foo</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Bar</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>id</td>
<td>name</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>you</td>
<td></td>
</tr>
</tbody>
</table>
Table Relationships

One-to-One

<table>
<thead>
<tr>
<th>invoices</th>
<th>orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id</td>
</tr>
<tr>
<td>order_id</td>
<td>name</td>
</tr>
</tbody>
</table>

class Invoice < ActiveRecord::Base
  belongs_to :order
end

class Order < ActiveRecord::Base
  has_one :invoice
end

One-to-Many

<table>
<thead>
<tr>
<th>line_items</th>
<th>orders</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>id</td>
</tr>
<tr>
<td>order_id</td>
<td>name</td>
</tr>
</tbody>
</table>

class LineItem < ActiveRecord::Base
  belongs_to :order
end

class Order < ActiveRecord::Base
  has_one :line_items
end
Table Relationships

Many-to-Many

categories

id
name
categories_products

category_id
product_id

products

id
name

class Category < ActiveRecord::Base
    has_and_belongs_to :products
end
class Product < ActiveRecord::Base
    has_and_belongs_to :categories
end
### One-to-Many

#### books

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>author</th>
<th>date</th>
<th>publisher_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Agile Web Development with Rails</td>
<td>Thomas, Dave</td>
<td>2005-11-17</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Programming Ruby</td>
<td>Thomas, Dave</td>
<td>2005-01-02</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Web Component Development with Zope 3</td>
<td>Weitershausen</td>
<td>2005-05-01</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The Pragmatic Bookshelf</td>
</tr>
<tr>
<td>2</td>
<td>Springer</td>
</tr>
</tbody>
</table>

CREATE TABLE `books` (
    `id` int(11) NOT NULL auto_increment,
    `title` varchar(100) NOT NULL,
    `author` varchar(50) NOT NULL,
    `date` date default '0000-00-00',
    `publisher_id` int(11) NOT NULL,
    PRIMARY KEY (`id`)
)

constraint fk_publisher foreign key (publisher_id) references publishers(id)

CREATE TABLE `publishers` (
    `id` int(11) NOT NULL auto_increment,
    `name` varchar(50) NOT NULL,
    PRIMARY KEY (`id`)
)
### One-to-Many

<table>
<thead>
<tr>
<th>id</th>
<th>title</th>
<th>author</th>
<th>date</th>
<th>publisher_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Agile Web Development with Rails</td>
<td>Thomas, Dave</td>
<td>2005-11-17</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Programming Ruby</td>
<td>Thomas, Dave</td>
<td>2005-01-02</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Web Component Development with Zope 3</td>
<td>Weitershausen</td>
<td>2005-05-01</td>
<td>2</td>
</tr>
</tbody>
</table>

```ruby
require "rubygems"
require_gem "activerecord"

ActiveRecord::Base.establish_connection(
  :adapter => "mysql",
  :host => "localhost",
  :database => "cs683BookStore_development",
  :username => "whitney")

class Book < ActiveRecord::Base
  belongs_to :publisher
end

class Publisher < ActiveRecord::Base
  has_many :books
end

springer = Publisher.create(:name => 'Springer')
ruby = Book.new
ruby.title = 'Web Component Development with Zope 3'
ruby.author = 'Weitershausen'
ruby.date = Time.now
ruby.publisher = springer
ruby.save

from_database = Book.find_by_author('Weitershausen')
puts from_database.publisher.name
```
Added Methods

class Book < ActiveRecord::Base
  belongs_to :publisher
end

Methods Added to Book

  publisher(force_reload=false)
publisher=(object)
build_product(attributes=\{\})
create_product(attribute=\{\})

class Publisher < ActiveRecord::Base
  has_many :books
end

Methods Added to Publisher

  books(force_reload=false)
books<<aBook
books.push(book1, ...)
books.delete(book1, book2, ...)
books.clear
books.find(options...)
books.build(attributes=\{\})
books.create(attributes=\{\})
books.size
books.empty?
Automatic Saving - One Way

```ruby
springer = Publisher.find_by_name('Springer')
networking = Book.new
networking.title = 'Fundametantal Networking in Java'
networking.date = Time.now
springer.books<<networking

Book networking is now saved in the database
```

```ruby
addison = Publisher.new
addison.name = 'Addison-Wesley'
ruby = Book.find_by_title('Programming Ruby')
ruby.publisher = addison
ruby.save

Now addison is saved in the database
ruby is updated
```

```ruby
addison = Publisher.new
addison.name = 'Addison-Wesley'
ruby = Book.find_by_title('Programming Ruby')
ruby.publisher = addison

Publisher addison is not saved
ruby not updated
```
Changing the Defaults

class Publisher < ActiveRecord::Base
  has_many :books,
    :class => 'PaperBook',
    :foreign_key => 'bk_id',
    :dependent => :destroy
end
CREATE TABLE `authors_books` (  `author_id` int(11) NOT NULL,  `book_id` int(11) NOT NULL,  PRIMARY KEY (`author_id`,`book_id`) ) TYPE=MyISAM

CREATE TABLE `authors` (  `id` int(11) NOT NULL auto_increment,  `first_name` varchar(30) NOT NULL,  `last_name` varchar(20) NOT NULL,  PRIMARY KEY (`id`) ) TYPE=MyISAM
class Book < ActiveRecord::Base
  belongs_to :publisher
  has_and_belongs_to_many :authors

  def to_s
    "#{title} by #{authors.collect {|a| a.to_s + ' '}}"
  end
end

class Publisher < ActiveRecord::Base
  has_many :book
end

thomas = Author.find_by_last_name('Thomas')
thomas.books.each {|book| puts book}
rails = Book.find(1)  # two authors
  rails.authors.each {|author| puts author}
ruby = Book.find(2)  # one author
  ruby.authors.each {|author| puts author}
Added Methods

class Author < ActiveRecord::Base
  has_and_belong_to_many :books
end

Methods Added to Author

books(force_reload=false)
books<<aBook
books.push_with_attributes(book1, ...)
books.delete(book1, book2,...)
books=objects
books.clear
books.empty?
books.size
books.find(id)
Concurrent Edits

Thread 1
a = Book.find(1)

a.title = "Foo"
a.save

Thread 2
b = Book.find(1)
b.title = "Bat"
b.save
Optimistic Locking

Version each row
   Add integer column 'lock_version' of zeros to table

Check version each update

Throw StaleObjectError if version changes before you update

CREATE TABLE `books` ( 
   `id` int(11) NOT NULL auto_increment, 
   `title` varchar(100) NOT NULL ,
   `date` date default '0000-00-00', 
   `publisher_id` int(11) NOT NULL, 
   `lock_version` int(11) default '0', 
   PRIMARY KEY (`id`) 
)

Thread 1
a = Book.find(1)
a.title = "Foo"
a.save  #now throws StaleObjectError

Thread 2
b = Book.find(1)
b.title = "Bat"
b.save
Thread 1
a = Book.find(1)
puts a.title

Thread2
b = Book.find(1)
b.title = "Foo"
b.save
puts b.title

Output
Agile Web Development

puts a.title
a.reload
puts a.title

Foo
Agile Web Development

Foo
Detecting when a save fails

```ruby
begin
  a = Book.find(1)
a.title = "Bar"
  a.save!
rescue RecordInvalid
  puts 'No Save'
end
```

```ruby
b = Book.find(1)
b.title = "Foo"
saved = b.save
if saved
  puts 'Successful save'
else
  puts 'No save'
end
```