# CS 683 Emerging Technologies
# Fall Semester, 2005
# Doc 24 Rails Database
# Nov 22, 2005

# 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

| SQL Type | Ruby Type |
|---|---|
| int, integer | Fixnum |
| decimal, numeric | Float |
| interval, date | Date |
| clob, blob, text | String |
| float, double | Float |
| char, varchar, string | String |
| datetime, time | Time |
| boolean | special handling needed |

# Normalization

First Normal Form

- Keep attributes single valued
- Put collections in separate table

**Books**

| id | Title | Author1 | Author2 |
|---|---|---|---|
| 1 | Foo | you | me |
| | | | |

**Books**

| id | title |
|---|---|
| 1 | Foo |
| | |

**Authors**

| id | name |
|---|---|
| 1 | you |
| 2 | me |

**Join Table**

| book_id | author_id |
|---|---|
| 1 | 1 |
| 1 | 2 |

# Normalization

Second Normal Form

All non-key attributes must be fully dependent on the entire primary key

Don't repeat data

**Books**

| id | title | publisher |
|---|---|---|
| 1 | Foo | you |
| 2 | Bar | you |

**Books**

| id | title | publisher_id |
|---|---|---|
| 1 | Foo | 1 |
| 2 | Bar | 1 |

**Publishers**

| id | name |
|---|---|
| 1 | you |

# Table Relationships

## One-to-One


```
class Invoice < ActiveRecord::Base
        belongs_to :order
end
```

```
class Order < ActiveRecord::Base
        has_one     :invoice
end
```

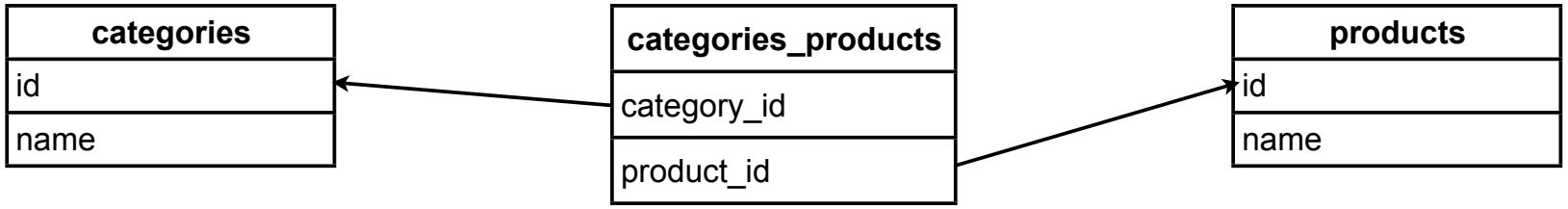
## One-to-Many


```
class LineItem < ActiveRecord::Base
        belongs_to :order
end
```

```
class Order < ActiveRecord::Base
        has_one     :line_items
end
```

# Table Relationships

## Many-to-Many




```
class Category < ActiveRecord::Base
        has_and_belongs_to    :products
end
```

```
class Product < ActiveRecord::Base
        has_and_belongs_to    :categories
end
```

# One-to-Many

**books**

| id | title | author | date | publisher_id |
|---|---|---|---|---|
| 1 | Agile Web Development with Rails | Thomas, Dave | 2005-11-17 | 1 |
| 2 | Programming Ruby | Thomas, Dave | 2005-01-02 | 1 |
| 3 | Web Component Development with Zope 3 | Weitershausen | 2005-05-01 | 2 |

*

1

**publishers**

| id | name |
|---|---|
| 1 | The Pragmatic Bookshelf |
| 2 | Springer |

```
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

**books**

| id | title | author | date | publisher_id |
|---|---|---|---|---|
| 1 | Agile Web Development with Rails | Thomas, Dave | 2005-11-17 | 1 |
| 2 | Programming Ruby | Thomas, Dave | 2005-01-02 | 1 |
| 3 | Web Component Development with Zope 3 | Weitershausen | 2005-05-01 | 2 |

*

1

**publishers**

| id | name |
|---|---|
| 1 | The Pragmatic Bookshelf |
| 2 | Springer |

```
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.emply?
```

# Automatic Saving - One Way

```
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
```

```
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
```

```
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',
          :forgeign_key => 'bk_id',
          :dependent => :destroy
end
```

# Many-to-Many

**books**

| id | title | date | publisher_id |
|---|---|---|---|
| 1 | Agile Web Development with Rails | 2005-11-17 | 1 |
| 2 | Programming Ruby | 2005-01-02 | 1 |
| 3 | Web Component Development with Zope 3 | 2005-05-01 | 2 |

*

1

**publishers**

| id | name |
|---|---|
| 1 | The Pragmatic Bookshelf |
| 2 | Springer |

**authors_books**

| book_id | author_id |
|---|---|
| 1 | 1 |
| 1 | 2 |
| 2 | 1 |
| 3 | 3 |

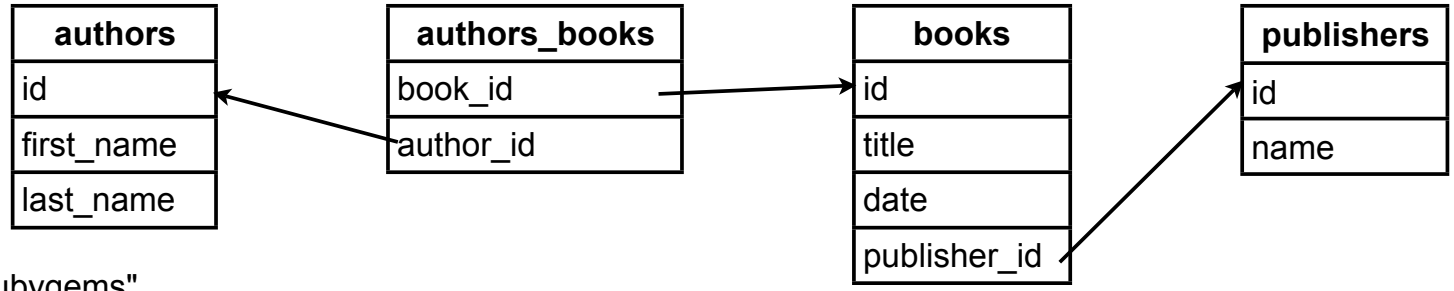
**authors**

| id | first_name | last_name |
|---|---|---|
| 1 | Dave | Thomas |
| 2 | David | Hansson |
| 3 | Philipp | Weiterhausen |

```
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
```

# Many-to-Many




```
require "rubygems"
require_gem "activerecord"

ActiveRecord::Base.establish_connection(
  etc)

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
```

```
class Author < ActiveRecord::Base
  has_and_belongs_to_many :books

  def to_s
    "#{first_name} #{last_name}"
  end
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_belongs_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** | **Thread 2** |
| --- | --- |
| a = Book.find(1) | |
| | b = Book.find(1) |
| | b.title = "Bat" |
| | b.save |
| a.title = "Foo" | |
| a.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** | **Thread 2** |
|---|---|
| a = Book.find(1) | |
| | b = Book.find(1) |
| | b.title = "Bat" |
| | b.save |
| a.title = "Foo" | |
| a.save  #now throws StaleObjectError | |

# reload

| Thread 1 | Thread2 | Output |
|---|---|---|
| a = Book.find(1) | | |
| puts a.title | | Agile Web Development |
| | b = Book.find(1) | |
| | b.title = "Foo" | |
| | b.save | |
| | puts b.title | Foo |
| puts a.title | | Agile Web Development |
| a.reload | | |
| puts a.title | | Foo |

## Detecting when a save fails

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

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