Hey... Is that your database crying ?

Blythe Dunham

blythe@snowgiraffe.com

http://snowgiraffe.com


Or maybe that was you?

Or maybe that was you?

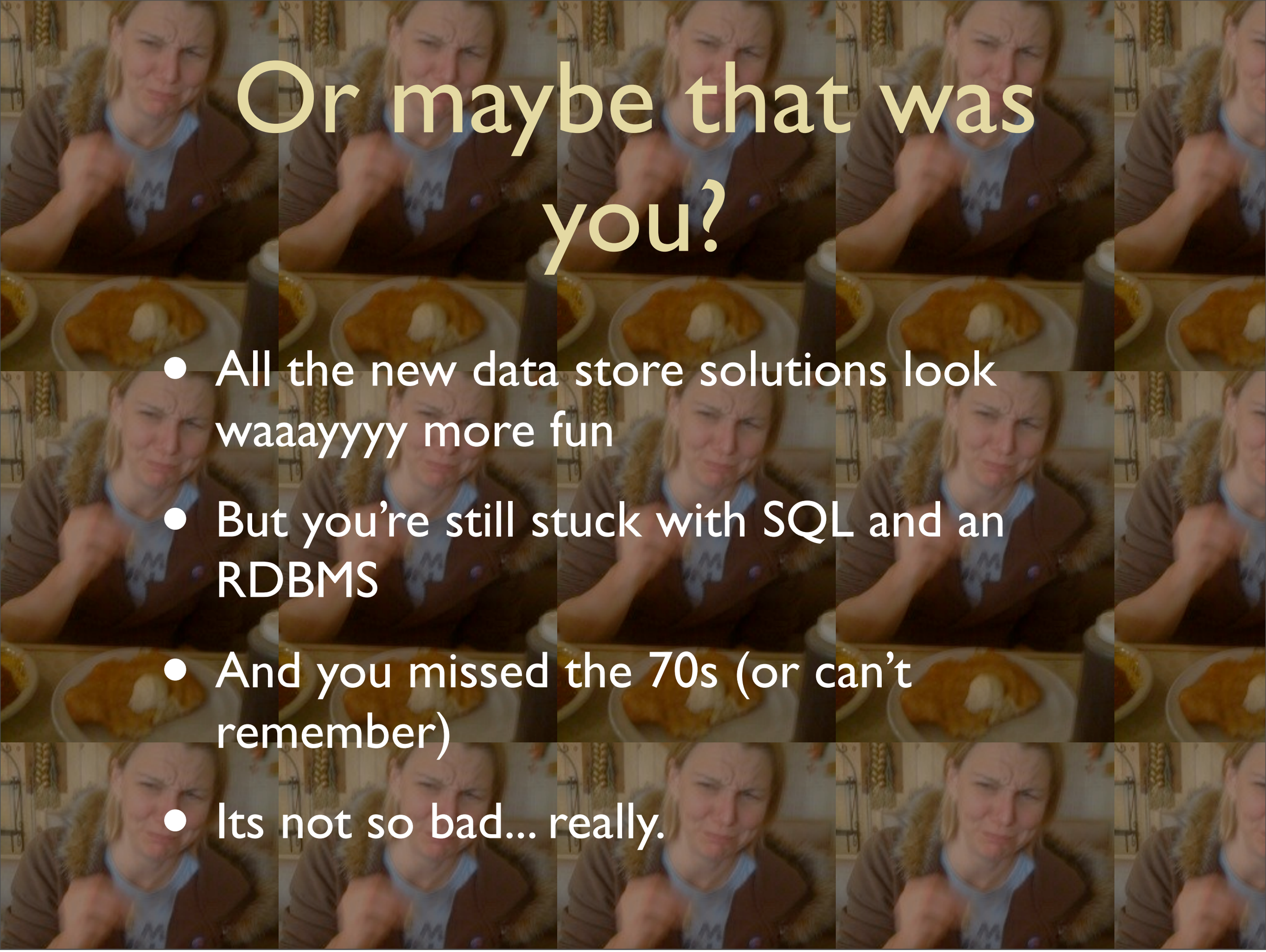
- All the new data store solutions look waaayyyy more fun

Or maybe that was you?

- All the new data store solutions look waaayyyy more fun
- But you're still stuck with SQL and an RDBMS

Or maybe that was you?

- All the new data store solutions look waaayyyy more fun
- But you're still stuck with SQL and an RDBMS
- And you missed the 70s (or can't remember)



Or maybe that was you?

- All the new data store solutions look waaayyyy more fun
- But you're still stuck with SQL and an RDBMS
- And you missed the 70s (or can't remember)
- Its not so bad... really.

Hey... Is that your database crying ?

Blythe Dunham

blythe@snowgiraffe.com

http://snowgiraffe.com

Welcome to the Boardshop

Model

Database Table

Migration Code

```
class Board < ActiveRecord::Base
end
```

```
CREATE TABLE `boards` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) DEFAULT NULL,
  `type` varchar(255) DEFAULT NULL,
  `manufacturer_id` int(11) DEFAULT NULL,
  `style` varchar(255) DEFAULT NULL,
  `description` text,
  `created_at` datetime DEFAULT NULL,
  `updated_at` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8
```

```
create_table :boards do |t|
  t.string :name
  t.string :type
  t.references :manufacturer
  t.string :style
  t.text :description

  t.timestamps
end
```

3 Migration Dos

#1 Not Null

```
t.string      :name,          :null => false
t.references  :manufacturer,  :null => false
```

#1 Not Null

```
t.string      :name,          :null => false
t.references  :manufacturer,  :null => false
```

- Basic data integrity

#1 Not Null

```
t.string      :name,          :null => false
t.references  :manufacturer,  :null => false
```

- Basic data integrity
- Faster joins

#2 Default Values

#2 Default Values

```
class Board < ActiveRecord::Base
  def initialize( args = nil )
    super( args )
    self.manufacturer ||= Manfacturer.first
  end

  def self.create_freeride( args = nil )
    new( args )
    self.style ||= 'freeride'
  end
end
```

#2 Default Values

#2 Default Values

```
t.string :style, :default => 'freeride'
t.text :description, :default => nil
```

#2 Default Values

```
t.string :style, :default => 'freeride'
t.text :description, :default => nil
```

- Cleans up the ORM code

#2 Default Values

```
t.string :style, :default => 'freeride'
t.text :description, :default => nil
```

- Cleans up the ORM code
- One less column to populate

ActiveRecord Defaults

```
class Board < ActiveRecord::Base
  default_value :name => 'freeride'
  defaults :manufacturer => lambda { |board|
    Manufacturer.first
  }
end
```

ActiveRecord Defaults

```
class Board < ActiveRecord::Base
  default_value :name => 'freeride'
  defaults :manufacturer => lambda { |board|
    Manufacturer.first
  }
end
```

Default Value For

```
script/plugin install git://github.com/FooBarWidget/default_value_for.git
```

ActiveRecord Defaults

```
script/plugin install
http://svn.viney.net.nz/things/rails/plugins/active_record_defaults
```

#3 Smaller Data Types

```
t.string  :name,  :limit => 64
t.integer :count, :limit => 2
```

#3 Smaller Data Types

```
t.string  :name,  :limit => 64
t.integer :count, :limit => 2
```

- String (varchar): 1 byte + length

#3 Smaller Data Types

```
t.string  :name,  :limit => 64
t.integer :count, :limit => 2
```

- String (varchar): 1 byte + length
- int(11) -> 11 is the display width, not the number of bytes

#3 Smaller Data Types

```
t.string  :name,  :limit => 64
t.integer :count, :limit => 2
```

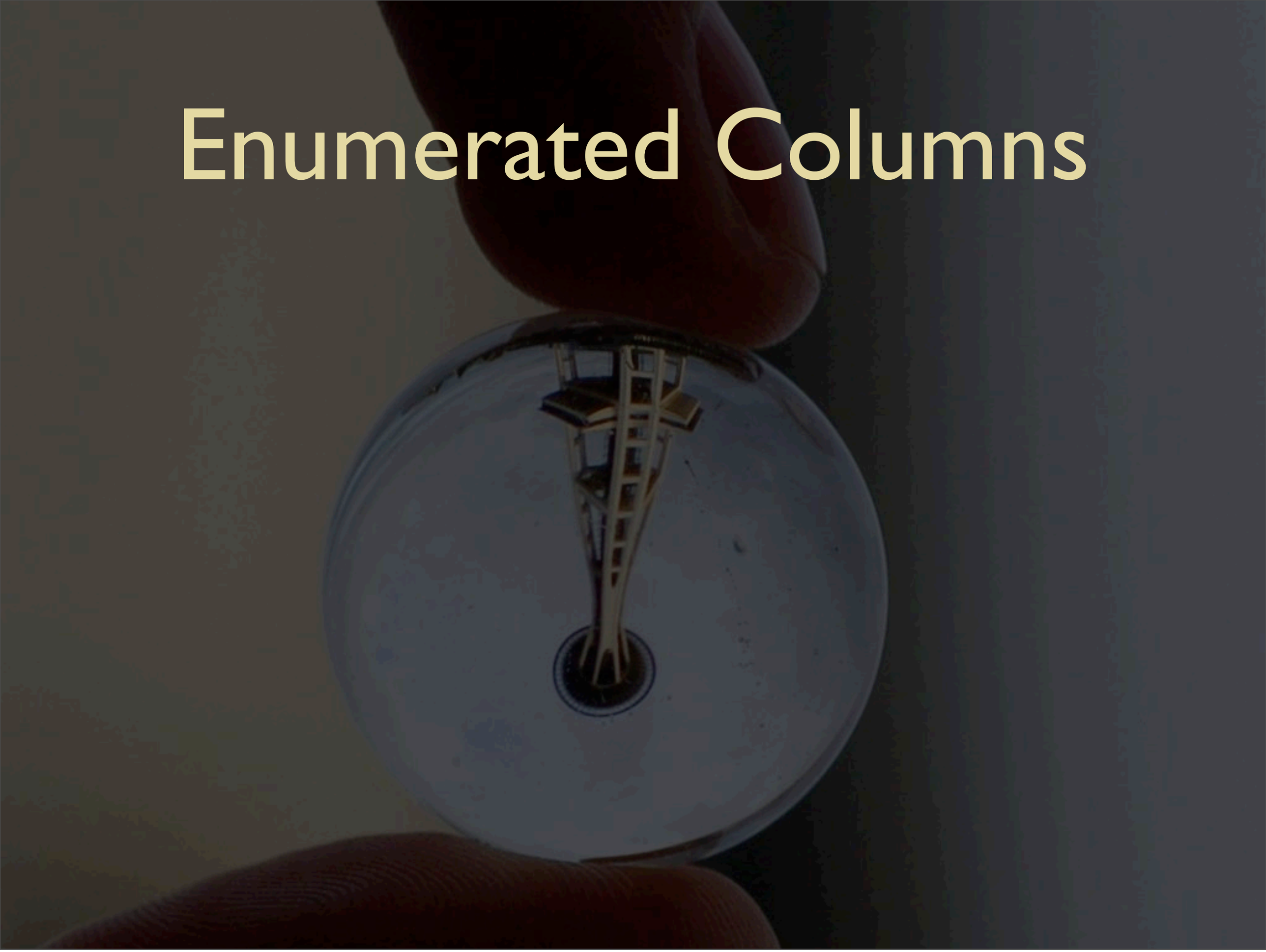
- String (varchar): 1 byte + length
- int(11) -> 11 is the display width, not the number of bytes
- Not such a huge deal (when not indexed)

#3 Smaller Data Types

```
t.string  :name,  :limit => 64
t.integer :count, :limit => 2
```

- String (varchar): 1 byte + length
- int(11) -> 11 is the display width, not the number of bytes
- Not such a huge deal (when not indexed)
- Most important: match data types on joined columns

Enumerated Columns



Enumerated Columns

```
t.string :style,  :default => 'freeride'
```

Enumerated Columns

```
t.string :style,  :default => 'freeride'
```

Enumerated Columns

```
t.string :style,   :default => 'freeride'
t.enum   :style,   :default => 'freeride',
         :limit => %w(freeride all_terrain freestyle)
```

Enumerated Columns

```
t.string :style,   :default => 'freeride'
t.enum   :style,   :default => 'freeride',
         :limit => %w(freeride all_terrain freestyle)
```

```
board.style == :freeride
```

Enumerated Columns

```
t.string :style,   :default => 'freeride'
t.enum   :style,   :default => 'freeride',
         :limit => %w(freeride all_terrain freestyle)

board.style == :freeride
```

- Enumerated Column Plugin

  script/plugin install http://rubyforge.org/var/svn/enum-column

Enumerated Columns

```
t.string :style,   :default => 'freeride'
t.enum   :style,   :default => 'freeride',
         :limit => %w(freeride all_terrain freestyle)

board.style == :freeride
```

- Enumerated Column Plugin

  script/plugin install http://rubyforge.org/var/svn/enum-column

- Enum column < 255 values uses one byte

Enumerated Columns

```
t.string :style,   :default => 'freeride'
t.enum   :style,   :default => 'freeride',
         :limit => %w(freeride all_terrain freestyle)

board.style == :freeride
```

- Enumerated Column Plugin

  script/plugin install http://rubyforge.org/var/svn/enum-column

- Enum column < 255 values uses one byte
- Try with Single Table Inheritance

FYI... Procedure analyse

```
mysql> select * from boards  PROCEDURE ANALYSE(10, 200000) \G
*************************** 1. row ***************************
             Field_name: boardshop_development.boards.id
              Min_value: 1
              Max_value: 2037203465
             Min_length: 1
             Max_length: 10
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 1018601733.0000
                    Std: 1018601732.0000
      Optimal_fieldtype: INT(10) UNSIGNED NOT NULL
*************************** 2. row ***************************
             Field_name: boardshop_development.boards.name
              Min_value: Custom X
              Max_value:Vinsen
             Min_length: 6
             Max_length: 8
       Empties_or_zeros: 0
                  Nulls: 0
Avg_value_or_avg_length: 7.3333
                    Std: NULL
      Optimal_fieldtype: ENUM('Custom X','Vinsen') NOT NULL
*************************** 3. row ***************************
```

2 Migration gotchas

#1 Migration Failures

#1 Migration Failures

#1 Migration Failures

- MySQL alter table cannot be rolled back

#1 Migration Failures

- MySQL alter table cannot be rolled back
- use create table :force => true do |t| ... end

#1 Migration Failures

- MySQL alter table cannot be rolled back
- use create table :force => true do |t| ... end
- in development:

#1 Migration Failures

- MySQL alter table cannot be rolled back
- use create table :force => true do |t| ... end
- in development:
  - comment out

#1 Migration Failures

- MySQL alter table cannot be rolled back
- use create table :force => true do |t| ... end
- in development:
  - comment out
  - script/console

#1 Migration Failures

- MySQL alter table cannot be rolled back
- use create table :force => true do |t| ... end
- in development:
  - comment out
  - script/console
- in production: rewrite migration with begin/rescue

#2 Custom DDL

#2 Custom DDL

Sometimes you need to write custom DDL

```
#case sensitive password (encrypted)
execute "ALTER TABLE `boards` ADD `permalink`
         varchar(25) character set latin1
         collate latin1_bin NOT NULL"
```

#2 Custom DDL

Sometimes you need to write custom DDL

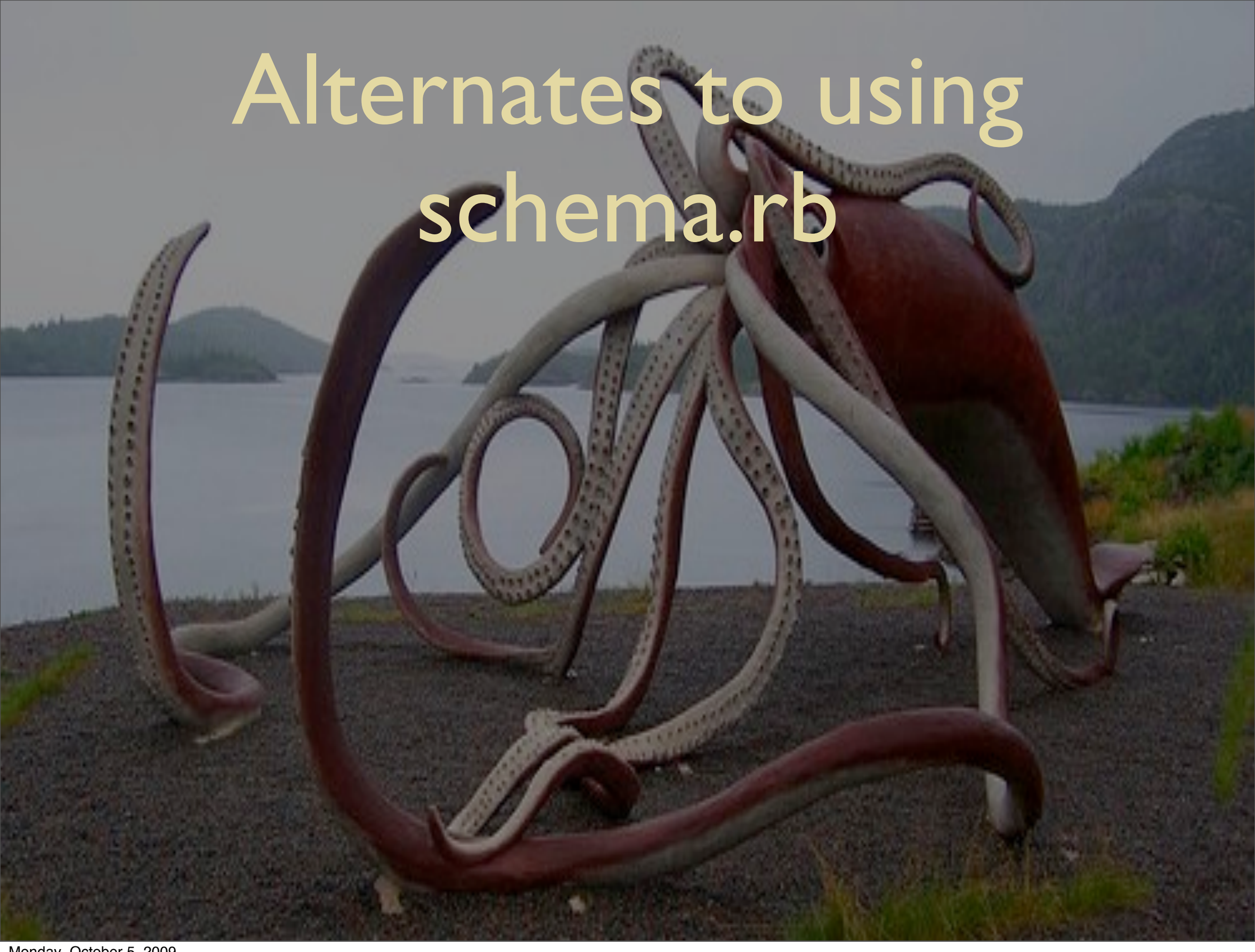
```
#case sensitive password (encrypted)
execute "ALTER TABLE `boards` ADD `permalink`
          varchar(25) character set latin1
          collate latin1_bin NOT NULL"
```

which can create invalid schema.rb files

```
t.string :permalink, :limit => 0
```

and rake db:schema:load fails

Alternates to using schema.rb



Alternates to using schema.rb

- Change the schema format in environment.rb

Alternates to using schema.rb

- Change the schema format in environment.rb

```
config.active_record.schema_format :sql
```

Alternates to using schema.rb

- Change the schema format in environment.rb

```
config.active_record.schema_format :sql
```

- Always migrate from 000

Alternates to using schema.rb

- Change the schema format in environment.rb

```
config.active_record.schema_format :sql
```

- Always migrate from 000
- Clone production/staging database

3 Reasons the RDBMS Cries When Talking to Rails

#1 Indexing

```
add_index :boards, :manufacturer_id,
          :name => 'board_manufacturer_index'
```

- speeds up queries on big tables
- huge cause of slowness

Composite Indexes

```
add_index :board,
          [:manufacturer_id, :manufacturer_type],
          :name => 'board_manufacturer_type_index'
```

Composite Indexes

```
add_index :board,
          [:manufacturer_id, :manufacturer_type],
          :name => 'board_manufacturer_type_index'
```

- Index across multiple columns

Composite Indexes

```
add_index :board,
          [:manufacturer_id, :manufacturer_type],
          :name => 'board_manufacturer_type_index'
```

- Index across multiple columns
- Order is important. Only the leftmost prefix is used

Composite Indexes

```
add_index :board,
          [:manufacturer_id, :manufacturer_type],
          :name => 'board_manufacturer_type_index'
```

- Index across multiple columns
- Order is important. Only the leftmost prefix is used
- Might use a composite with Polymorphism

Unique Keys

```
add_index :boards, :name,
          :unique => true,
          :name => 'uk_boards_name'
```

Unique Keys

```
add_index :boards, :name,
          :unique => true,
          :name => 'uk_boards_name'
```

ORM checks do not guaranty data integrity!!!

```
class Boards < ActiveRecord::Base
  validates_uniqueness_of :name
end
```

Fasten your Foreign Key Seatbelts

Fasten your Foreign Key Seatbelts

- Provides referential integrity

Fasten your Foreign Key Seatbelts

- Provides referential integrity
- Without can easily orphan records with update_all, delete_all

Fasten your Foreign Key Seatbelts

- Provides referential integrity
- Without can easily orphan records with update_all, delete_all
- Easier and faster to delete related records

Fasten your Foreign Key Seatbelts

- Provides referential integrity
- Without can easily orphan records with update_all, delete_all
- Easier and faster to delete related records
- Index for free

The Rails Way

```
class Manufacturer < ActiveRecord::Base
  has_many :boards, :dependent => :nullify
end
```

The Rails Way

```
class Manufacturer < ActiveRecord::Base
  has_many :boards, :dependent => :nullify
end
```

`:dependent` option database equivalents (MySQL):

The Rails Way

```
class Manufacturer < ActiveRecord::Base
  has_many :boards, :dependent => :nullify
end
```

`:dependent` option database equivalents (MySQL):

- :nullify => ON DELETE SET NULL

The Rails Way

```
class Manufacturer < ActiveRecord::Base
  has_many :boards, :dependent => :nullify
end
```

`:dependent` option database equivalents (MySQL):

- :nullify => ON DELETE SET NULL
- :delete_all => ON DELETE CASCADE

The Rails Way

```
class Manufacturer < ActiveRecord::Base
  has_many :boards, :dependent => :nullify
end
```

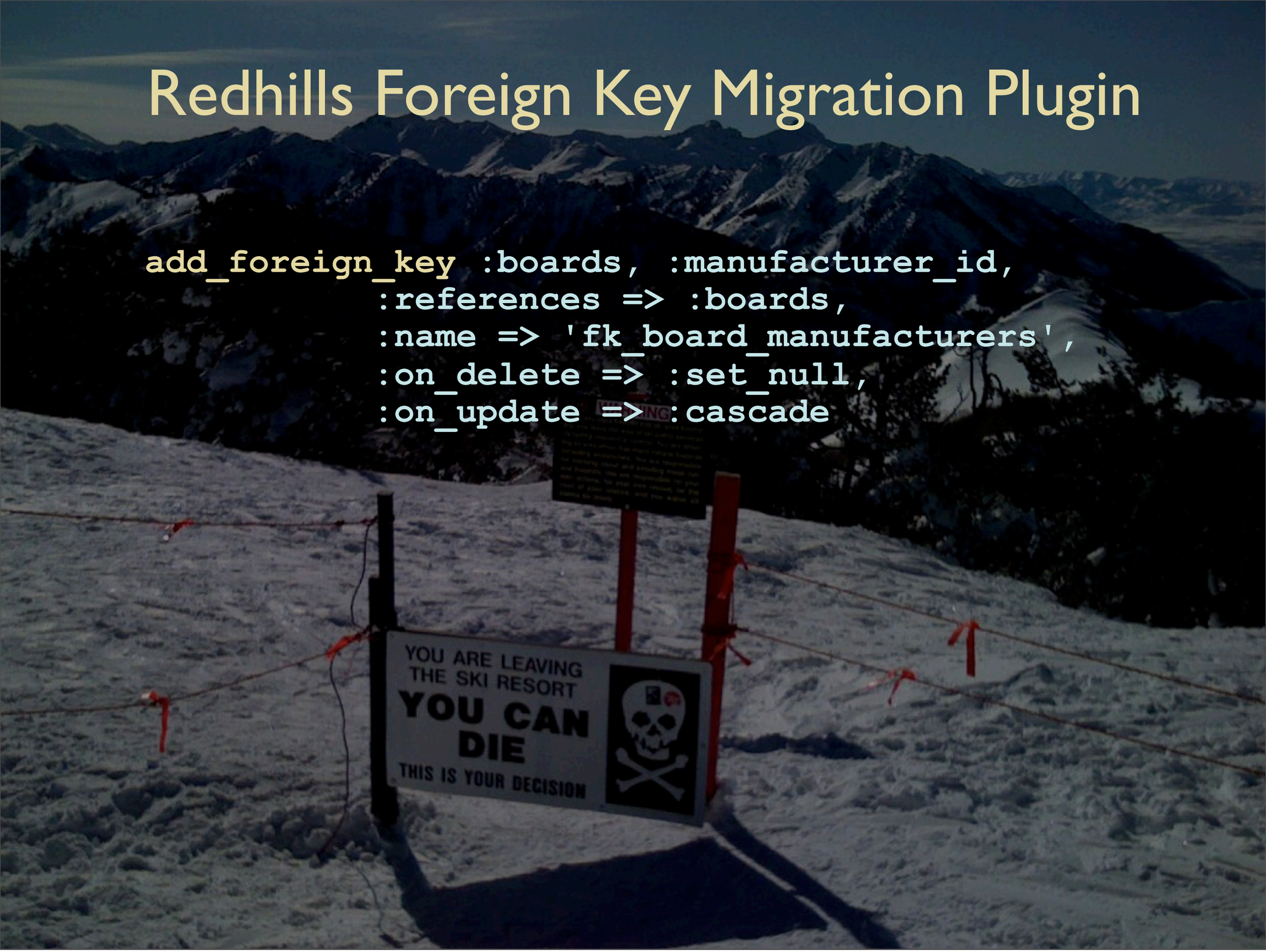
`:dependent` option database equivalents (MySQL):

- :nullify => ON DELETE SET NULL
- :delete_all => ON DELETE CASCADE
- :destroy => No SQL equivalent. Every association is instantiated and and callbacks are executed before destruction

Redhills Foreign Key Migration Plugin

Redhills Foreign Key Migration Plugin

```
add_foreign_key :boards, :manufacturer_id,
                :references => :boards,
                :name => 'fk_board_manufacturers',
                :on_delete => :set_null,
                :on_update => :cascade
```



Redhills Foreign Key Migration Plugin

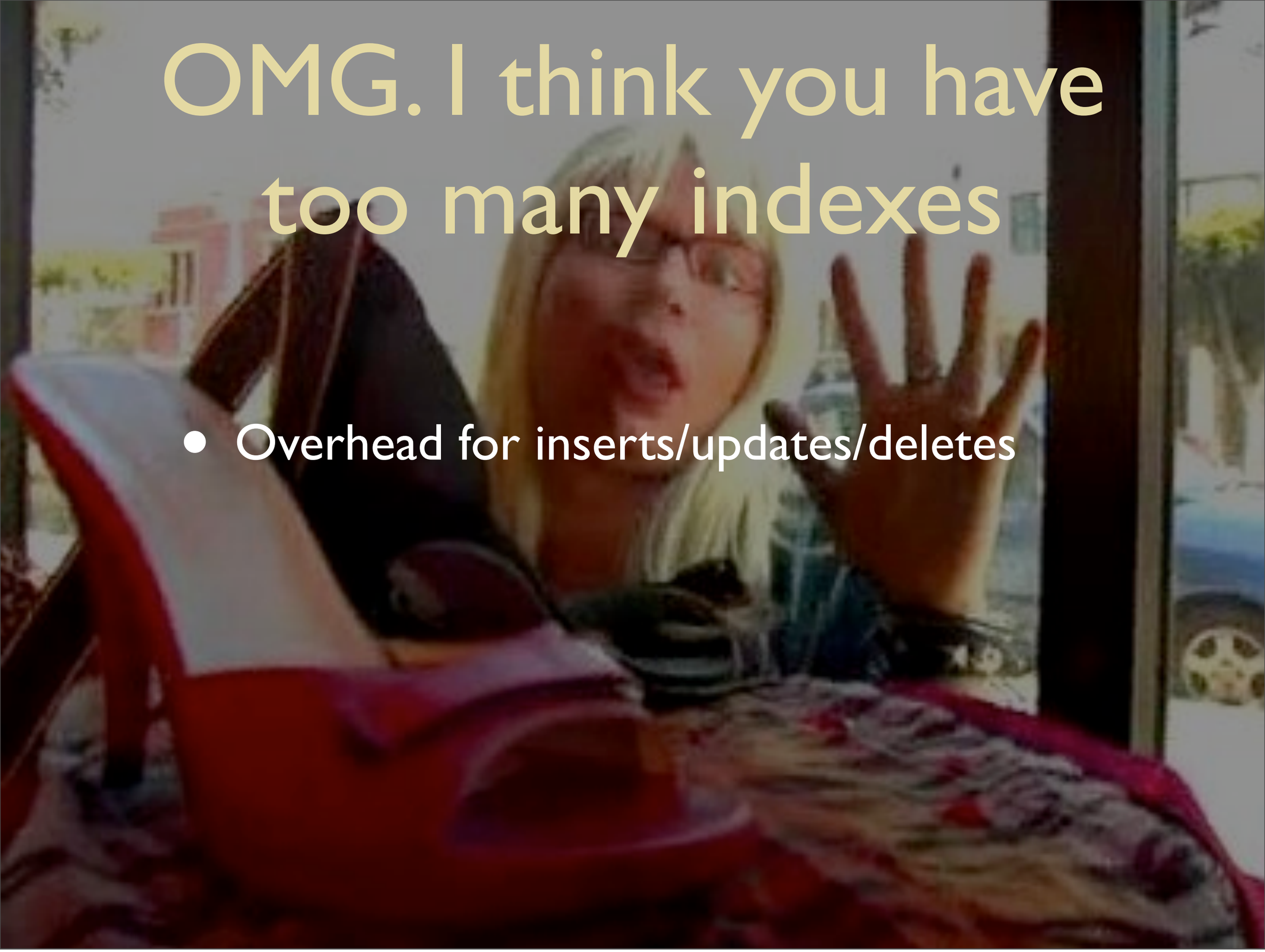
```
add_foreign_key :boards, :manufacturer_id,
                :references => :boards,
                :name => 'fk_board_manufacturers',
                :on_delete => :set_null,
                :on_update => :cascade
```

```
create_table :boards do |t|
  t.integer     :manufacturer_id,
                :on_delete => :cascade,
                :null => false

  t.references  :manufacturers,
                :on_delete => :set_null,
                :default => nil

end
```


OMG. I think you have
too many indexes


OMG. I think you have too many indexes
• Overhead for inserts/updates/deletes

OMG. I think you have too many indexes

- Overhead for inserts/updates/deletes
- Indexes do takes up a bit of space

OMG. I think you have too many indexes

- Overhead for inserts/updates/deletes
- Indexes do takes up a bit of space
- Too many indexes can confuse the database

OMG. I think you have too many indexes

- Overhead for inserts/updates/deletes
- Indexes do takes up a bit of space
- Too many indexes can confuse the database
- Indexes are not always used by the database

General Guidelines

Unless the table is small:

General Guidelines

Unless the table is small:

- Index Foreign Relationships

General Guidelines

Unless the table is small:

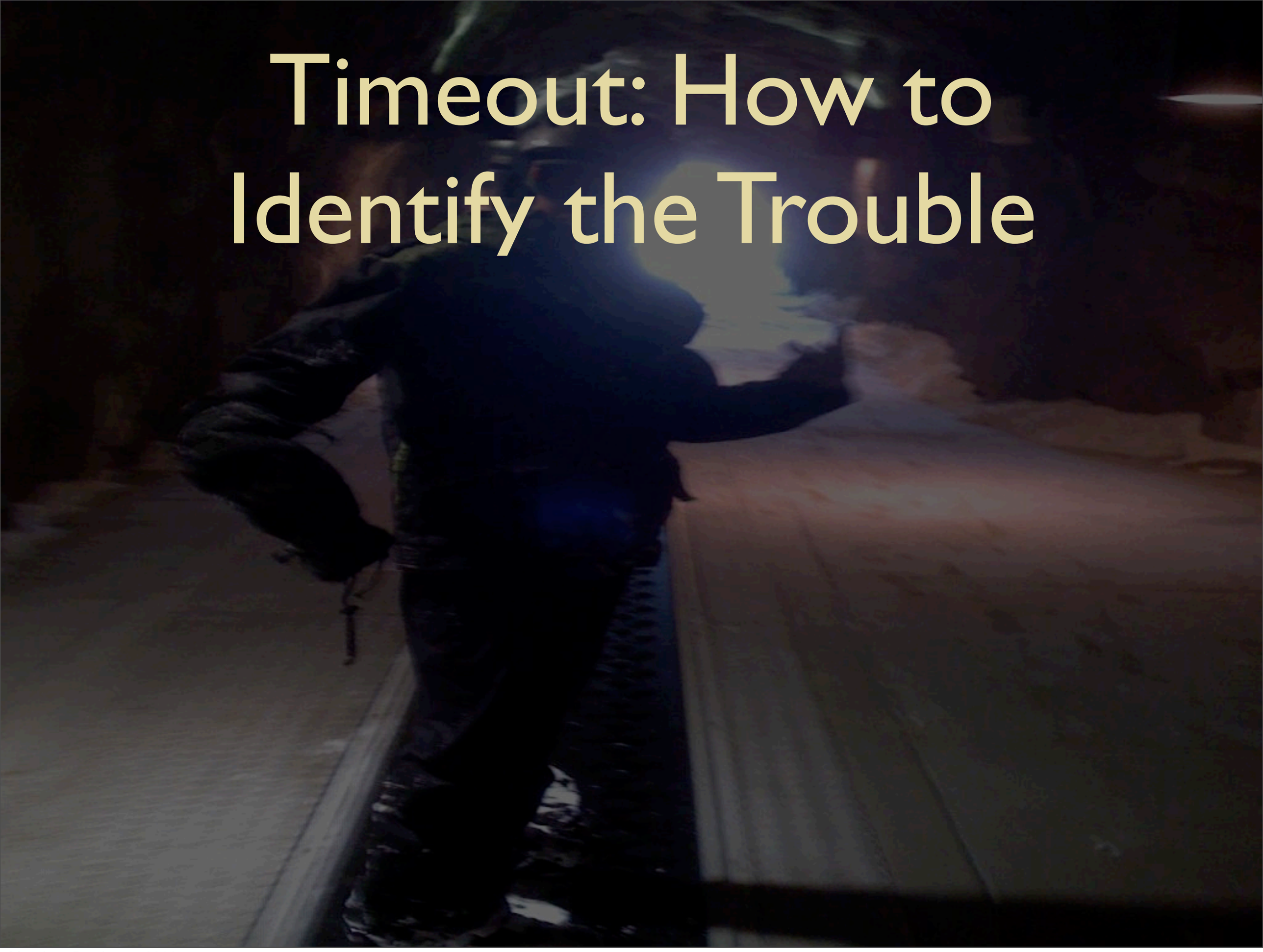
- Index Foreign Relationships
- Unique Columns

General Guidelines

Unless the table is small:

- Index Foreign Relationships
- Unique Columns
- Columns that are queried on or sorted on

Timeout: How to Identify the Trouble



Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB

Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots

Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots
- Check out the slow query log

Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots
- Check out the slow query log
- See the picture with FiveRuns, NewRelic, rawk, rails_analyzer_tools, query analyzer

Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots
  - Check out the slow query log
  - See the picture with FiveRuns, NewRelic, rawk, rails_analyzer_tools, query analyzer
- Pinpoint trouble

Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots
  - Check out the slow query log
  - See the picture with FiveRuns, NewRelic, rawk, rails_analyzer_tools, query analyzer
- Pinpoint trouble
  - Examine Single Actions (enable caching)

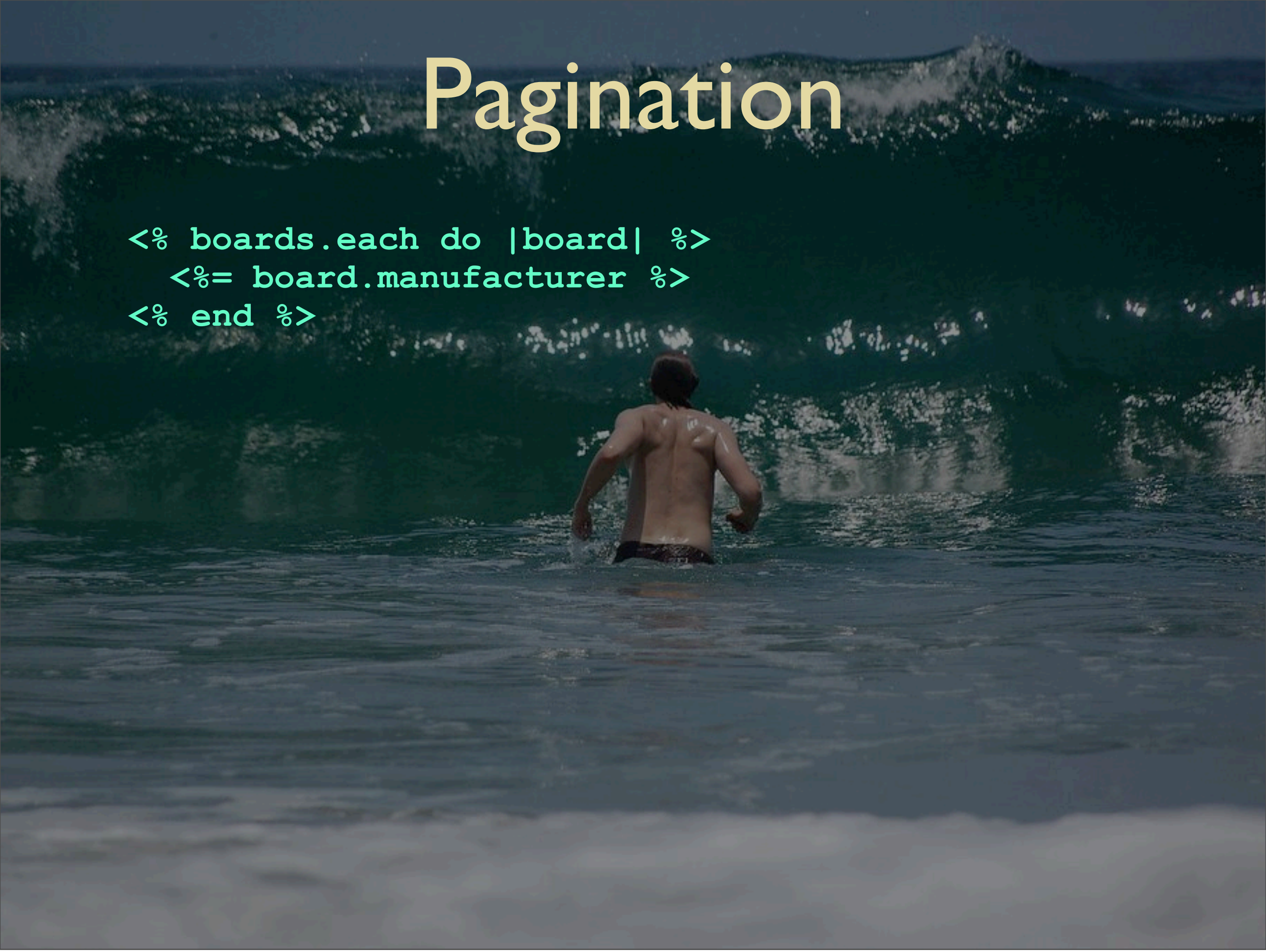
Timeout: How to Identify the Trouble

- Familiarize yourself with Rails interaction with DB
- Identify trouble spots
  - Check out the slow query log
  - See the picture with FiveRuns, NewRelic, rawk, rails_analyzer_tools, query analyzer
- Pinpoint trouble
  - Examine Single Actions (enable caching)
  - Ruby Prof http://ruby-prof.rubyforge.org/

#2 Big Queries on Giant Columns

Pagination

```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>
```



Pagination

```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>
```

Will Paginate Gem/Plugin
http://github.com/mislav/will_paginate

```
#controller
@boards = Board.paginate :page => params[:page]
```

Pagination

```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>

<%= will_paginate @boards %>
```

Will Paginate Gem/Plugin
http://github.com/mislav/will_paginate

```
#controller
@boards = Board.paginate :page => params[:page]
```

Pagination

```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>

<%= will_paginate @boards %>
```

Will Paginate Gem/Plugin
http://github.com/mislav/will_paginate

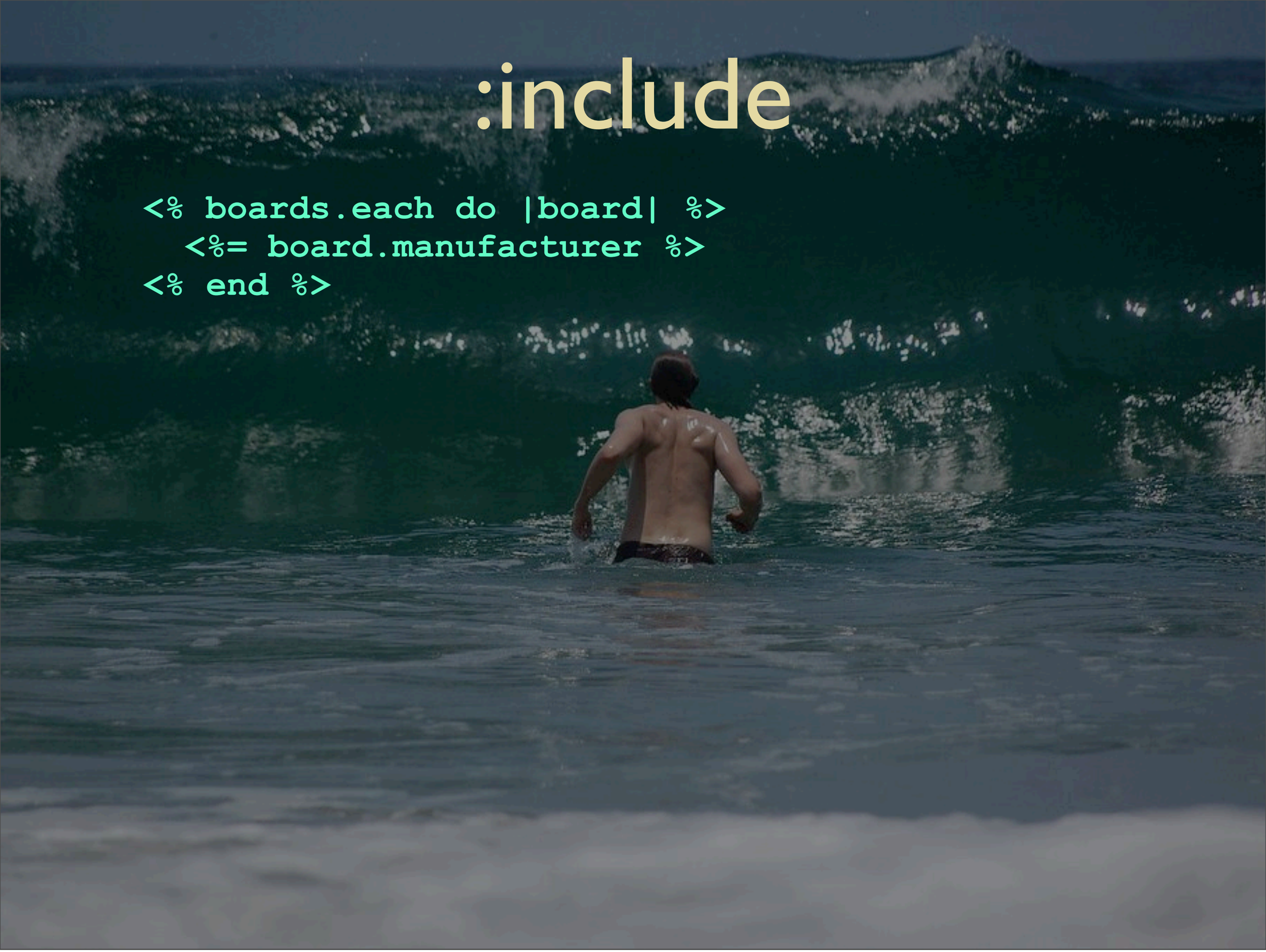
```
#controller
@boards = Board.paginate :page => params[:page]
```

Rails Pagination

```
Board.find_each(:batch_size => 2) do |board|
  board.surf!
end
```

:include

```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>
```



:include

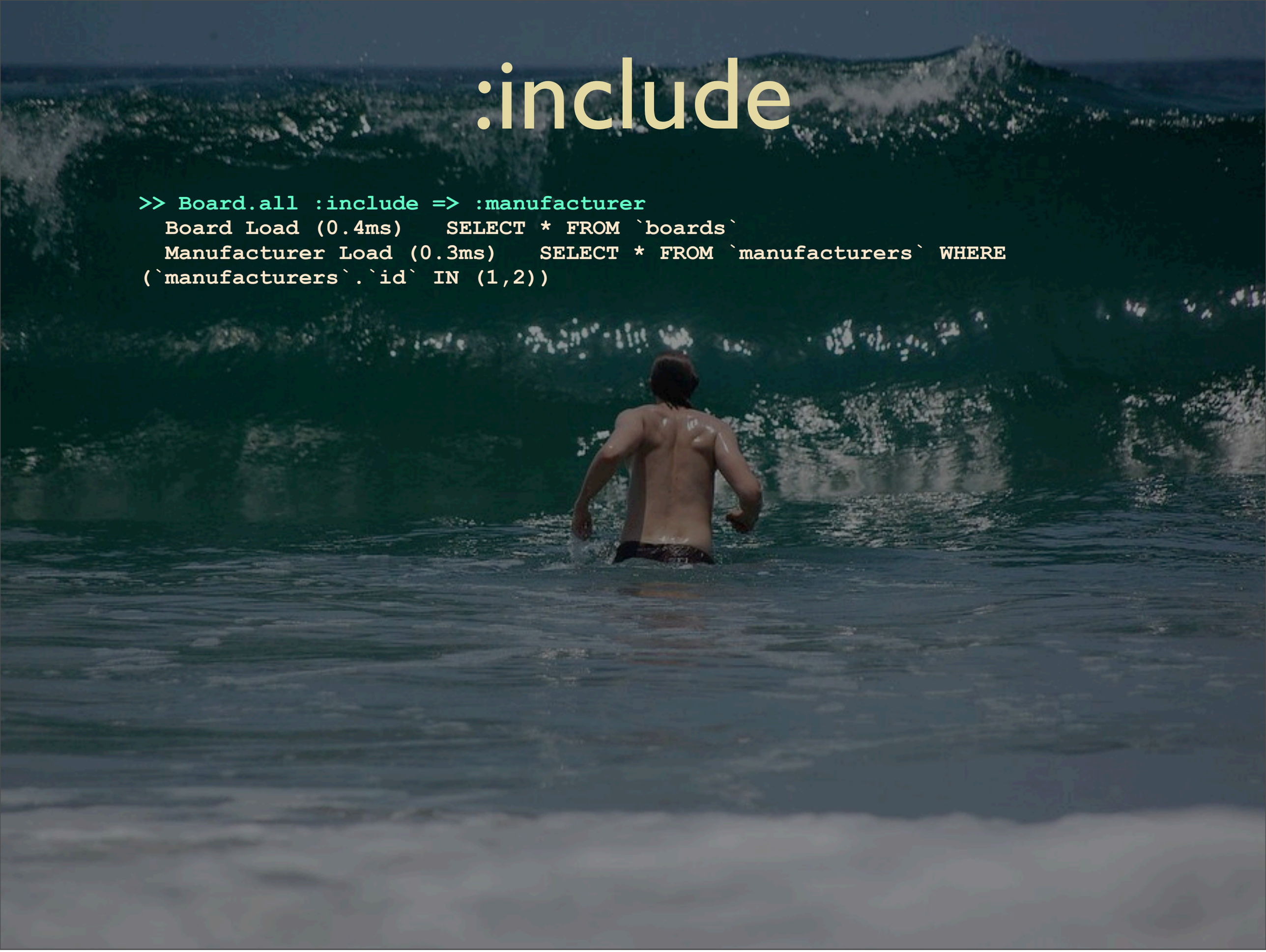
```
<% boards.each do |board| %>
  <%= board.manufacturer %>
<% end %>
```

Eager load references that are called often

```
@boards = Board.all(
   :conditions => ...,
   :include => [:manufacturer, :pro_riders]
)
```

:include

```
>> Board.all :include => :manufacturer
  Board Load (0.4ms)   SELECT * FROM `boards`
  Manufacturer Load (0.3ms)   SELECT * FROM `manufacturers` WHERE
(`manufacturers`.`id` IN (1,2))
```



:include

```
>> Board.all :include => :manufacturer
  Board Load (0.4ms)   SELECT * FROM `boards`
  Manufacturer Load (0.3ms)   SELECT * FROM `manufacturers` WHERE
(`manufacturers`.`id` IN (1,2))
```

Conditions can generate huge queries

```
>> Board.all :include => :manufacturer,
  :conditions => {'manufacturers.name' => 'Burton'}
  Board Load Including Associations (1.2ms)   SELECT `boards`.`id` AS
t0_r0, `boards`.`name` AS t0_r1, `boards`.`type` AS t0_r2,
`boards`.`manufacturer_id` AS t0_r3, `boards`.`style` AS t0_r4,
`boards`.`description` AS t0_r5, `boards`.`created_at` AS t0_r6,
`boards`.`updated_at` AS t0_r7, `manufacturers`.`id` AS t1_r0,
`manufacturers`.`name` AS t1_r1, `manufacturers`.`description` AS t1_r2,
`manufacturers`.`created_at` AS t1_r3, `manufacturers`.`updated_at` AS
t1_r4 FROM `boards` LEFT OUTER JOIN `manufacturers` ON `manufacturers`.id =
`boards`.manufacturer_id WHERE (`manufacturers`.`name` = 'Burton')
```

:select

```
Board.all :select => 'DISTINCT style'

class Manufacturer < ActiveRecord::Base
  has_many :boards,
           :select => 'boards.name, boards.id'
end
```

:select

```
Board.all :select => 'DISTINCT style'

class Manufacturer < ActiveRecord::Base
  has_many :boards,
           :select => 'boards.name, boards.id'
end

class Board < ActiveRecord::Base
  named_scope :simple,
              {:select => 'id, name' }
end
```

:select

```
Board.all :select => 'DISTINCT style'

class Manufacturer < ActiveRecord::Base
  has_many :boards,
           :select => 'boards.name, boards.id'
end

class Board < ActiveRecord::Base
  named_scope :simple,
              {:select => 'id, name' }
end
```

- Data is retrieved faster when fewer columns are selected

:select

```
Board.all :select => 'DISTINCT style'

class Manufacturer < ActiveRecord::Base
  has_many :boards,
           :select => 'boards.name, boards.id'
end

class Board < ActiveRecord::Base
  named_scope :simple,
              {:select => 'id, name' }
end
```

- Data is retrieved faster when fewer columns are selected
- Error if you reference a column not selected

:select doesn't work with :include

```
Board.all :include => :manufacturer,
          :select => 'name, manufacturer.name'
```

:select doesn't work with :include

```
Board.all :include => :manufacturer,
          :select => 'name, manufacturer.name'
```

- Use :joins (inner) instead of :include
- Eager loading plugin
- Redefine Relationships

Redefine Relationships

Redefine Relationships

Queries text or large string columns can really slow things down

```
create_table :boards do |t|
  t.text        :description
  ...
end
```

Redefine Relationships

Queries text or large string columns can really slow things down

```
create_table :boards do |t|
  t.text        :description
  ...
end
```

```
create_table :board_descriptions do |t|
  t.text        :description
  t.references  :board
end
```

Redefine Relationships

```
create_table :board_descriptions do |t|
  t.text       :description
  t.references :board
end
```

Redefine Relationships

```
create_table :board_descriptions do |t|
  t.text       :description
  t.references :board
end
```

```
class Board < ActiveRecord::Base
```

```
end
```

Redefine Relationships

```
create_table :board_descriptions do |t|
  t.text       :description
  t.references :board
end
```

```
class Board < ActiveRecord::Base
  has_one  :board_description

end
```

Redefine Relationships

```
create_table :board_descriptions do |t|
  t.text       :description
  t.references :board
end
```

```
class Board < ActiveRecord::Base
  has_one  :board_description

  delegate :description,
           :to => :board_description,
           :prefix => false


end
```

Redefine Relationships

```
create_table :board_descriptions do |t|
  t.text       :description
  t.references :board
end
```

```
class Board < ActiveRecord::Base
  has_one  :board_description

  delegate :description,
           :to => :board_description,
           :prefix => false

  named_scope :with_description,
              { :include => :board_description }

end
```

#3 Inserting eleventy billion records one by one

#3 Inserting eleventy billion records one by one

Insert each snowboard one by one

```
Board.create!(:name => 'Custom X',
              :style => 'all_terrain')
Board.create!(:name => 'Vinsen',
              :style => 'freeride')
```

and so on....

Import

```
Board.import([
  Board.new(:name => 'Custom X',
            :style => 'all_terrain'),
  Board.new(:name => 'Vinsen',
            :style => 'freeride')
])
```

AR-Extensions:
sudo gem install ar-extensions

Avoid instantiating models

```
columns = [ :name, :style ]
values = [
  ['Custom X', 'all-terrain'],
  ['Vinsen', 'freeride']
]
options = {
  :validate => false,
  :timestamps => false
}
Board.import columns, values, options
```

Update or ignore existing records

- Update or ignore existing records using SQL
- Only works with unique key constraints

Update or ignore existing records

```
Board.import(
   columns,
   values,
  :on_duplicate_key_update => [:style, :updated_at]
)
```

- Update or ignore existing records using SQL
- Only works with unique key constraints

Update or ignore existing records

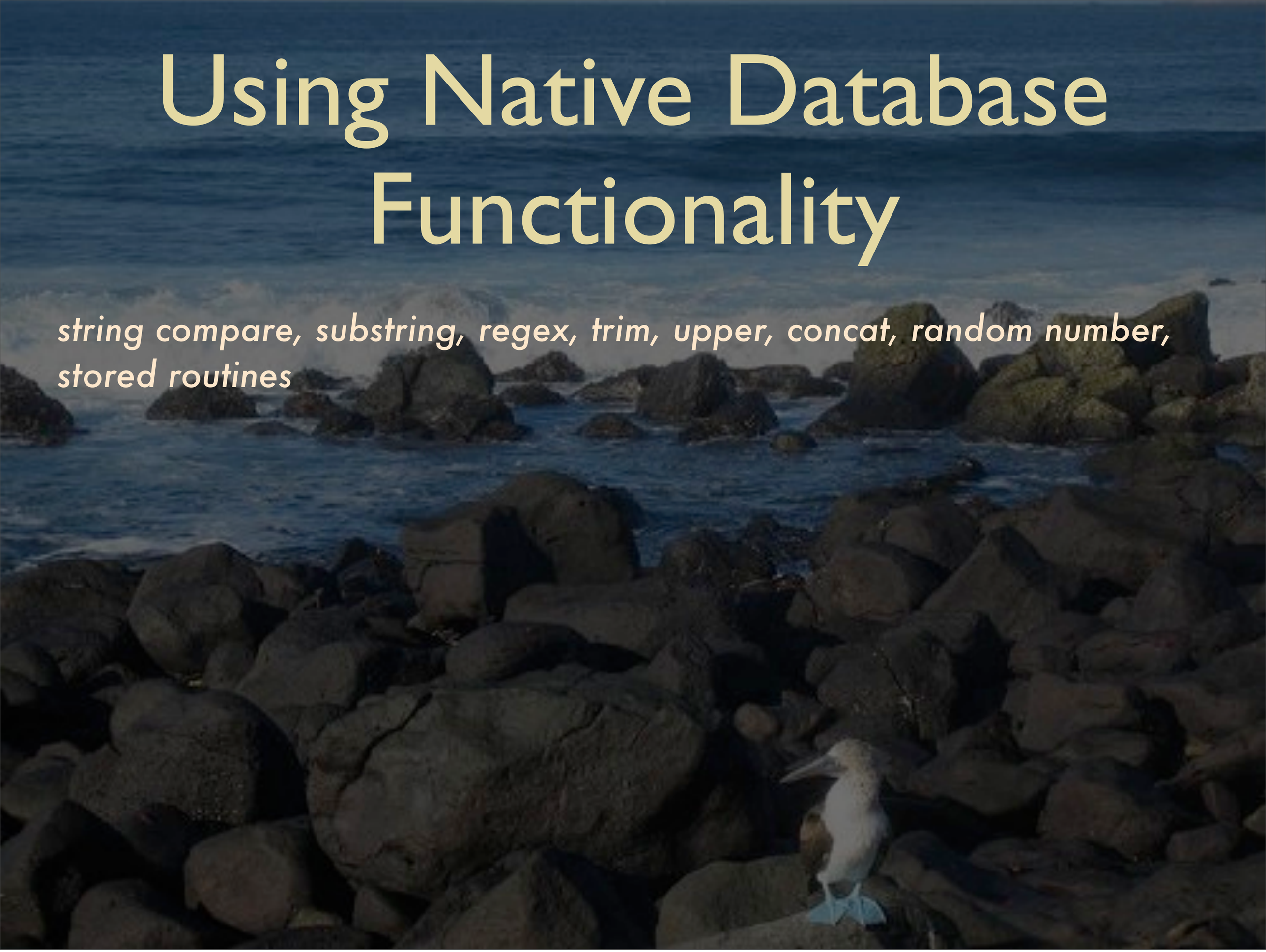
```
Board.import(
   columns,
   values,
  :on_duplicate_key_update => [:style, :updated_at]
)
```

```
Board.create!({:name => 'fun'}, :ignore => true)
```

- Update or ignore existing records using SQL
- Only works with unique key constraints

Using Native Database Functionality

string compare, substring, regex, trim, upper, concat, random number, stored routines



Using Native Database Functionality

string compare, substring, regex, trim, upper, concat, random number, stored routines

```
#slow
Board.find_each do |board|
  board.update_attribute(:name, “fun_#{board.name}”)
end
```

Using Native Database Functionality

string compare, substring, regex, trim, upper, concat, random number, stored routines

```
#slow
Board.find_each do |board|
  board.update_attribute(:name, “fun_#{board.name}”)
end

#better
Board.update_all([“name = CONCAT(?, name)”, ‘fun_’])
```

3 More Useful Tricks

#1 Deadlock Retry

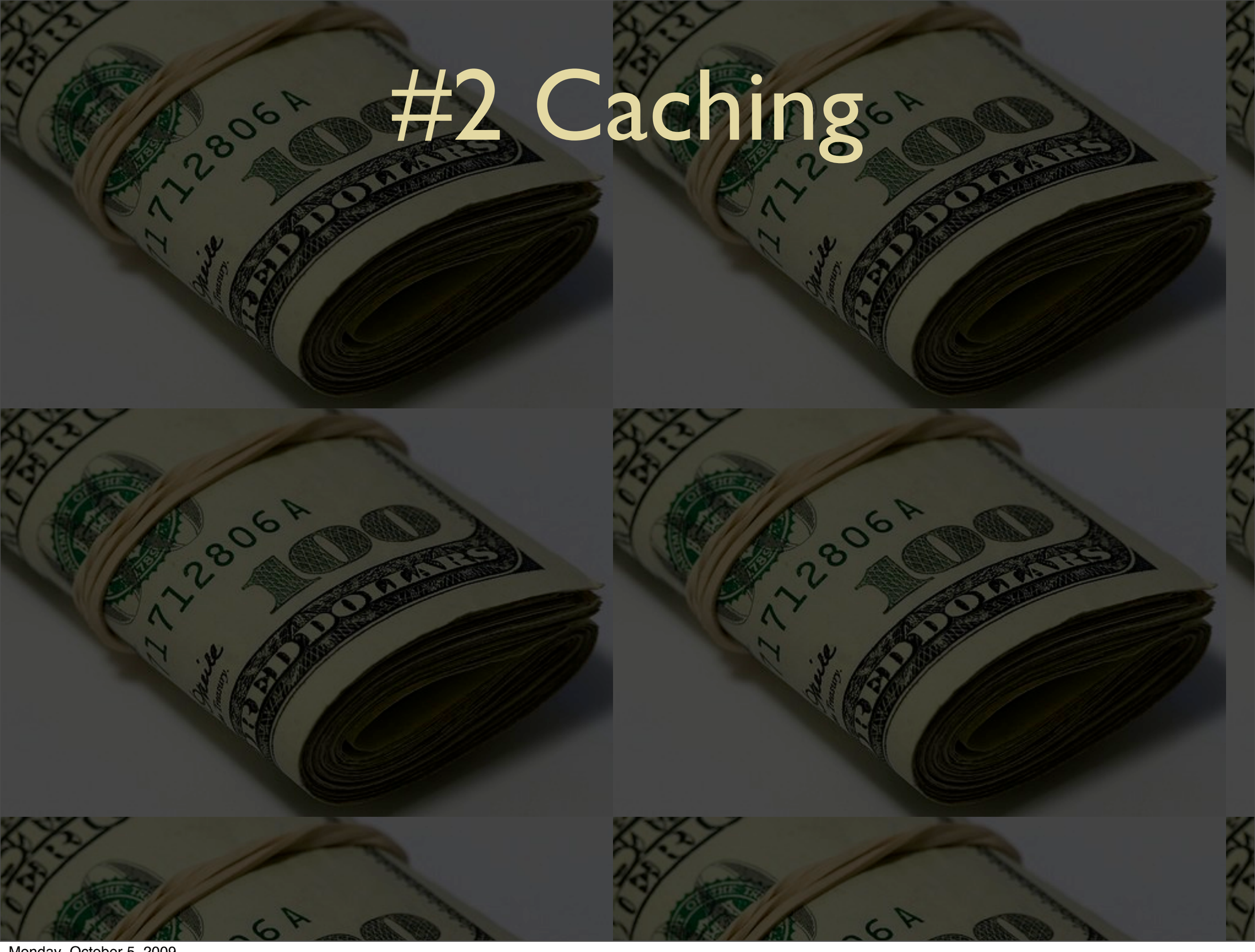
http://github.com/mperham/deadlock_retry

#1 Deadlock Retry

http://github.com/mperham/deadlock_retry

- Bigger queries mean more deadlock
- Tries 3 times before failing

#2 Caching



#2 Caching

- help from query cache

#2 Caching

- help from query cache
- memcache

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching
- ORM caches

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching
- ORM caches
- Enumerate By

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching
- ORM caches
- Enumerate By

script/plugin installl git://github.com/pluginaweek/enumerate_by

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching
- ORM caches
  - Enumerate By

    script/plugin installl git://github.com/pluginaweek/enumerate_by
  - StaticRecordCache

#2 Caching

- help from query cache
- memcache
- page/action/fragment caching
- ORM caches
  - Enumerate By

    script/plugin installl git://github.com/pluginaweek/enumerate_by
  - StaticRecordCache

    script/plugin installl git://github.com/pluginaweek/enumerate_by

#3 Replication

#3 Replication

- Reduce contention by load balancing requests
- Write to the master, read from the slave

It all started with masochism...

Read from the slave, write to the master

Masochism - The original

http://github.com/technoweenie/masochism

Master Slave Adapter - works with connection pooling

http://github.com/mauricio/master_slave_adapter

Data Fabric - sharding options

http://github.com/mperham/data_fabric

Multi DB - load balancing, nesting

http://github.com/schoefmax/multi_db

AR-Extensions http://www.continuousthinking.com/tags/arext

Eager Loading http://www.snowgiraffe.com/tech/?p=329

Will Paginate http://github.com/mislav/will_paginate/tree/master

Deadlock Retry http://agilewebdevelopment.com/plugins/deadlock_retry

Static Record Cache http://github.com/blythedunham/static_record_cache/tree/maste

Default Value For http://github.com/FooBarWidget/default_value_for.git

ActiveRecord Defaults http://svn.viney.net.nz/things/rails/plugins/active_record_defau

Enumerate By http://github.com/pluginaweek/enumerate_by

Enum Column http://rubyforge.org/var/svn/enum-column

Data Integrity http://github.com/blythedunham/rails_devs_for_data_integrity

geese migration: http://www.flickr.com/photos/napix/3513353546/
shopping cart: http://www.flickr.com/photos/16029748@N00/3523449828/
seatbelt: http://www.flickr.com/photos/uncleboatshoes/8701484/
big surf: http://www.flickr.com/search/?q=giant+surf
big wave: http://www.flickr.com/photos/thewentworths/1384082774/
big surf2: http://www.flickr.com/photos/robygbh/3874411103/in/set-72157601637705625/
omg shoes: http://2.bp.blogspot.com/_uB-BSH_BgAs/SUbo7ZY8iUI/AAAAAAAAC18/sAf9mM-Zmrg/s400/shoes+kelly.jpg
beach flame: http://www.flickr.com/photos/twozdai/3506547419/
surf boards stacked: http://www.flickr.com/photos/bluxx999/2594658129/
colorful surf boards: http://www.flickr.com/photos/pinksherbet/173466867/
1980's burton snowboards: http://www.flickr.com/photos/21532294@N07/2425729083/
march of the penguins: http://www.flickr.com/photos/brynj/38666954/in/photostream/
lotsa penguins: http://www.flickr.com/photos/slowloris/87793361/
wave: http://www.flickr.com/photos/adrasteia9/3527880636
columns and bench: http://www.flickr.com/photos/adrasteia9/4068068/
crusing with the top down: http://www.flickr.com/photos/adrasteia9/16542569
surf shark: http://www.guzer.com/pictures/surfing_shark_scare.jpg
redefining relationships: http://www.flickr.com/photos/adrasteia9/2734990410