Ruby Object Database
github.com/apohllo/rod

Aleksander Pohl
apohllo@o2.pl
apohllo.pl

Krakow Ruby Users Group

2nd, August 2011

Agenda

Introduction

Details

Help needed!

Agenda

Introduction

Details

Help needed!

Few words 'bout me

- PhD student at AGH-UST
- Assistant lecturer at Jagiellonian University
- Author of the Polish Introduction to Ruby
  `apohllo.pl/dydaktyka/ruby/intro`
- Maintainer of the Polish Rails Guides
  `apohllo.pl/guides/index.html`
- Mostly interested in Natural Language Processing (with Ruby)
  `github.com/apohllo/rod`
  `github.com/apohllo/polish-spec`
  `github.com/apohllo/rlp-grammar`
  `github.com/apohllo/rlp-semantics`
  `github.com/apohllo/rlp-corpus`

What ROD is not?

- is **not** relational database (MySQL, Postgresql, SQLite)
- is **not** normalized
- is **not** object-relational mapper (ActiveRecord, Sequel, DataMapper)
- is **not** database server (Internet/Unix socket communication)
- is **not** in-memory database (Redis)
- is **not** document database (MongoDB)
- is **not** prevalence database (Madelein, Prevayler)

What is ROD?

- (Ruby) object database
- partially based on the network database model
- uses (almost) the same address space as the Ruby process
- will use Oracle Berkeley DB as a backend
- designed for fast access to data which doesn't fit into memory, but should be available on one node
- a *kind* of data warehouse

Why?

Think of a library (with books) or an encyclopedia. You don't need all the information, but you might need any piece of it.

Object oriented access to:

- **text corpora**
  hundreds of millions of text segments, with interlinks
- rich natural **language dictionaries** for NLP
  millions of text forms, hundreds of millions of relationships
- e.g. WordNet is stored as Berkeley DB

Agenda

Introduction

Details

Help needed!

Show me some code! – database and class definitions

```ruby
class MyDatabase < Rod::Database
end

class Model < Rod::Model
  database_class MyDatabase
end

class User < Model
  field :name, :string
  field :surname, :string, :index => :btree,
    :sort => lambda {|s1,s2| p2 <=> p1 }
  field :age, :integer
  has_one :account
  has_many :files

  validates_presence_of :name, :surname
end
```

Class definitions

```ruby
class Account < Model
  field :email, :string
  field :login, :string, :index => :hash
  field :password, :string

  validates_presence_of :email, :login, :password
end

class File < Model
  field :title, :string, :index => :btree
  field :data, :string

  validates_presence_of :title
end
```

Object creation and storage

```
MyDatabase.create_database("data")
user = User.new(:name => 'Fred',
                :surname => 'Smith',
                :age => 22)
account = Account.new(:email => "fred@smith.org",
                      :login => "fred",
                      :password => "password")
file1 = File.new(:title => "Lady Gaga video")
file2.data = "0012220001..."
file2 = File.new(:title => "Pink Floyd video")
file2.data = "0012220001..."
user.account = account
user.files << file1
user.files << file2

user.store
account.store
file1.store
file2.store
MyDatabase.close_database
```

Object accessing

```ruby
MyDatabase.open_database("data")
User.each do |user|
  puts "Name: #{user.name} surname: #{user.surname}"
  puts "login: #{user.account.login} e-mail: #{user.account.email}"
  user.files.each do |file|
    puts "File: #{file.title}"
  end
end

User[0]
# gives first user
User.find_by_surname("Smith")
# gives Fred
User.find_all_by_surname("Smith")
# gives [Fred]
File[0].user
# won't work - the data is not normalized
```

Requirements

- Ruby 1.9
- RubyInline
- english gem
- ActiveModel
- Berkeley DB

Features/future (1/2)

- nice Ruby interface which mimicks Active Record
- Ruby-to-C on-the-fly translation based on mmap and RubyInline (will be replaced with Berkeley DB)
- append of the database (new objects, new elements in plural associations)
- full CRUD (TBD)
- optimized for (reading) speed
- ActiveModel validations
- ActiveModel dirty tracking (TBD)

Features/future (2/2)

- weak reference collections for easy memory reclaims
- hash/btree indices for short start-up times and key-sort support (TBD)
- compatibility check of library version
- compatibility check of data model
- autogeneration of model (based on the database metadata)
- automatic model migrations (addition/removal of properties so far)
- tested with Cucumber

Agenda

Introduction

Details

Help needed!

Help needed

- 52 issues on Github – much more to come
- design – review the architecture on project wiki
- testing – running of different OS
- testing – writing Cucumber specs
- this is really fun!
- a perfect candidate ;-)
  - will have knowledge of Berkeley DB
  - will have knowledge of Ruby C API
  - will be experienced in writing concurrent Ruby apps

Thank you!