Ruby Object Database

github.com/apohllo/rod
Aleksander Pohl

What ROD is not?

- is **not** relational database
- is **not** normalized
- is **not** object-relational mapper
- is **not** database server
- is **not** in-memory database
- is **not** document database
- is **not** prevalence database

What is ROD?

- fast access to data which doesn't fit into memory
- strict schema
- object-oriented interface
- based on the network database model
- same address space as Ruby
- Oracle Berkeley DB for indexing
- a *kind* of data warehouse

Why?

Object oriented access to:

- **text corpora**: hundreds of millions of text segments
- **natural language dictionaries**: millions of text forms, hundreds of millions of relationships

DDL

```
class User < Model
  field :name, :string
  field :surname, :string, :index => :hash
  field :age, :integer
  has_one :account
  has_many :files

  validates_presence_of :name, :surname
end
```

DML

```
MyDatabase.instance.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")
file1.data = "0012220001..."
user.account = account
user.files << file1

user.store
account.store
file1.store
MyDatabase.instance.close_database

MyDatabase.instance.open_database("data")
User.each{|u| puts "#{u.name} #{u.surname}" }
User.first.account.login
User.first.files.first.title
User.find_by_surname("Smith")
User.find_all_by_surname("Smith")
File.first.user
MyDatabase.instance.close_database
```

Tests

T.	I.	Sqlite[s]	ROD[s]	Speedup
1	0	33.65	11.32	x 2.97
1	1	33.66	**0.99**	x 34.00
1	2	33.54	1.08	x 31.05
2	0	42.48	2.79	x 15.22
2	1	41.87	**1.27**	x 32.97
2	2	42.47	**1.27**	x 33.44

```
form = WordForm.find_by_value(word)
if form && form.flexemes.count == 1 &&
    form.flexemes.first.word_forms.count > 1 &&
    form.flexemes.first.taggings.
    any?{|t| t.tags.map(&:value).include?(:noun)}
  count += 1
end
```


BDAS