

# Ruby DBMS Duct Tape

By Stefan Penner



# Stefan Penner

[iamstef.net](http://iamstef.net)

[github.com/stefanpenner](https://github.com/stefanpenner)

[@stefanpenner](#)

[irc: iamstef](#)

# Game Plan

- Database
- the Stack
- DBI
- DO
- Active Record
- Datamapper
- Sequel

# a Database is

a system intended to organize, store, retrieve large amounts of data easily.

Excel can store data



Excel is easy.

=

OMG, I Excel spreadsheet, shared drive,  
150+ users, 5 provinces!



deadlock much?

all databases are not equal!

databases evolve.

systems become legacy

SO....

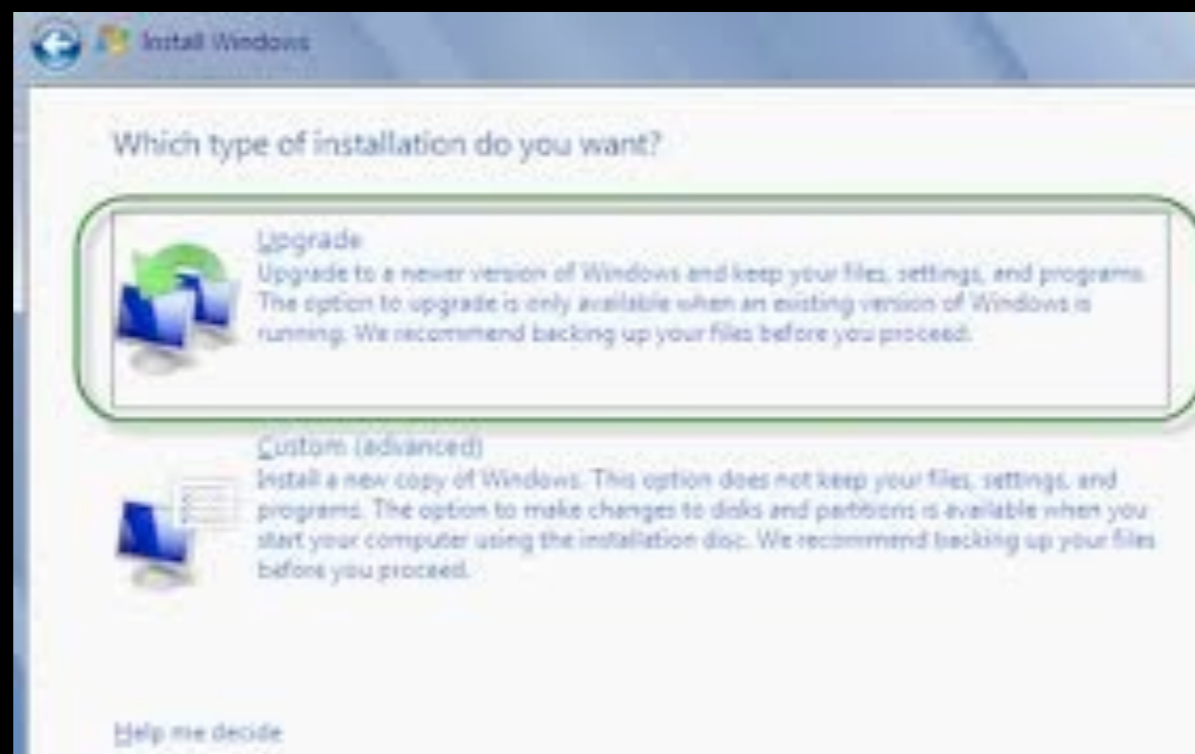
Lets Upgrade!



and funny pictures

# I once upgraded Windows once...

This will be easy...



# But...

- Vendor doesn't exist anymore
- no actual upgrade path
- no data integrity
- it costs money?

# So How do we Database?

- DBI
- Active Record
- DataMapper
- Sequel
- ODBC
- jRuby (JDBC)

# DBI

A Lightweight centralized API to database manipulation.

- adapter exists for ruby, your golden
- oldschool syntax
- no ORM

<http://ruby-dbi.rubyforge.org/>

Supports

- MYSQL
- ODBC
- Postgres
- SQLite(3)

# DBI Syntax Example

```
require 'dbi'

# Connect to a database, old style
dbh = DBI.connect('DBI:Mysql:test', 'testuser', 'testpwd')

# Insert some rows, use placeholders
1.upto(13) do |i|
  sql = "insert into simple01 (SongName, SongLength_s) VALUES (?, ?)"
  dbh.do(sql, "Song #{i}", "#{i*10}")
end

# Select all rows from simple01
sth = dbh.prepare('select * from simple01')
sth.execute

# Print out each row
while row=sth.fetch do |row|
  p row
end
```

# Active Record

Design pattern coined by Martin Fowler in  
“Patterns of enterprise application architecture”.

Also, the name of Ruby on Rails’s default ORM.

- Lots of Power
- Lots of Opinion
- Might fight with you

<https://github.com/rails/rails/tree/master/activerecord>

# Active Record Syntax (Raw)

```
require 'active_record'
```

```
ActiveRecord::Base.establish_connection({  
  :adapter => 'mysql',  
  :database => 'test_database',  
  :username => 'tester',  
  :password => 'test22'  
})
```

```
connection = ActiveRecord::Base.connection
```

```
connection.tables
```

```
> ['users', 'products', 'ducks', 'oranges']
```

```
users = []
```

```
connection.execute('SELECT * FROM users').each_hash do |user|  
  users << user  
end
```

```
users
```

```
> .... array of users, each user as a hash.
```

```
users.first
```

```
> { :id => 1, :username => 'stefan', :password => 'is_super_secure' }
```

# Active Record Syntax (ORM)

```
require 'active_record'
```

```
ActiveRecord::Base.establish_connection({  
  :adapter => 'mysql',  
  :database => 'test_database',  
  :username => 'tester',  
  :password => 'test22'  
})
```

```
# class <camel_case_singular_table_name> < ActiveRecord::Base  
class User < ActiveRecord::Base  
  # if the table's name does not fit convention, it can be manually overridden.  
  # table_name :users_table  
end
```

```
User.first
```

```
> #<User id: 2, :name => "stefan", :password => "is_super_secure" >
```

```
User.find(2)
```

```
> #<User id: 2, :name => "stefan", :password => "is_super_secure" >
```

```
User.where(:name => "stefan")
```

```
> #<User id: 2, :name => "stefan", :password => "is_super_secure" >
```

```
# Nathans Talk goes here.
```

# Sequel

Elegant Full featured database toolkit for ruby.

- Supports a metric Fuckton
- DSL
- ORM

<http://sequel.rubyforge.org/>

## Supports

- ADO, Amalgalite, DataObjects, DB2, DBI, Firebird, Informix, JDBC, MySQL, Mysql2, ODBC, OpenBase, Oracle, PostgreSQL, SQLite3, and Swift

# Sequel Example

```
require "sequel"

# connect to an in-memory database
DB = Sequel.sqlite()

# create an items table
DB.create_table :items do
  primary_key :id
  String :name
  Float :price
end

# create a dataset from the items table
items = DB[:items]

# populate the table
items.insert(:name => 'abc', :price => rand * 100)
items.insert(:name => 'def', :price => rand * 100)
items.insert(:name => 'ghi', :price => rand * 100)

# print out the number of records
puts "Item count: #{items.count}"

# print out the average price
puts "The average price is: #{items.avg(:price)}"
```

# Case Studies

# Clarion Top Speed based application for client X (April 2010)

## Database Stats

©1995

1350 Tables

200+ attributes

no referential Integrity

ODBC driver via Yandex

Windows



## Goal

Full Read/Write for new web ERP

## Deadline

1 Week ago



# Access Databases Sync With Rails Applications (Jan 2011)

## Goal

Read Access database from ruby, for syncing purposes

## Solution 1 (Multi-platform, but costs money)

jRuby

JDBC

HXTT's Access\_JDBC30.jar (<http://www.hxtt.com/>)

Sequel

## Solution 2 (Windows Only)

Ruby

ADO via WIN32OLE

Sequel

## Solution 3 (Mac Only, read only)

Ruby

ODBC

ActualTech's Mac only ODBC MS Access driver

Sequel

# Solution I (Multi-platform, but costs money)

```
require 'sequel'
require 'sequel/adapters/jdbc'
require 'sequel/jdbc_hxtt_adapter'
require 'Access_JDBC30.jar' # jRuby fun

DB = Sequel.connect("jdbc:access://///path/to/file.mdb")
puts DB.tables

> [:table_1, :table_2, :table_3, :table_4, ... ]
```

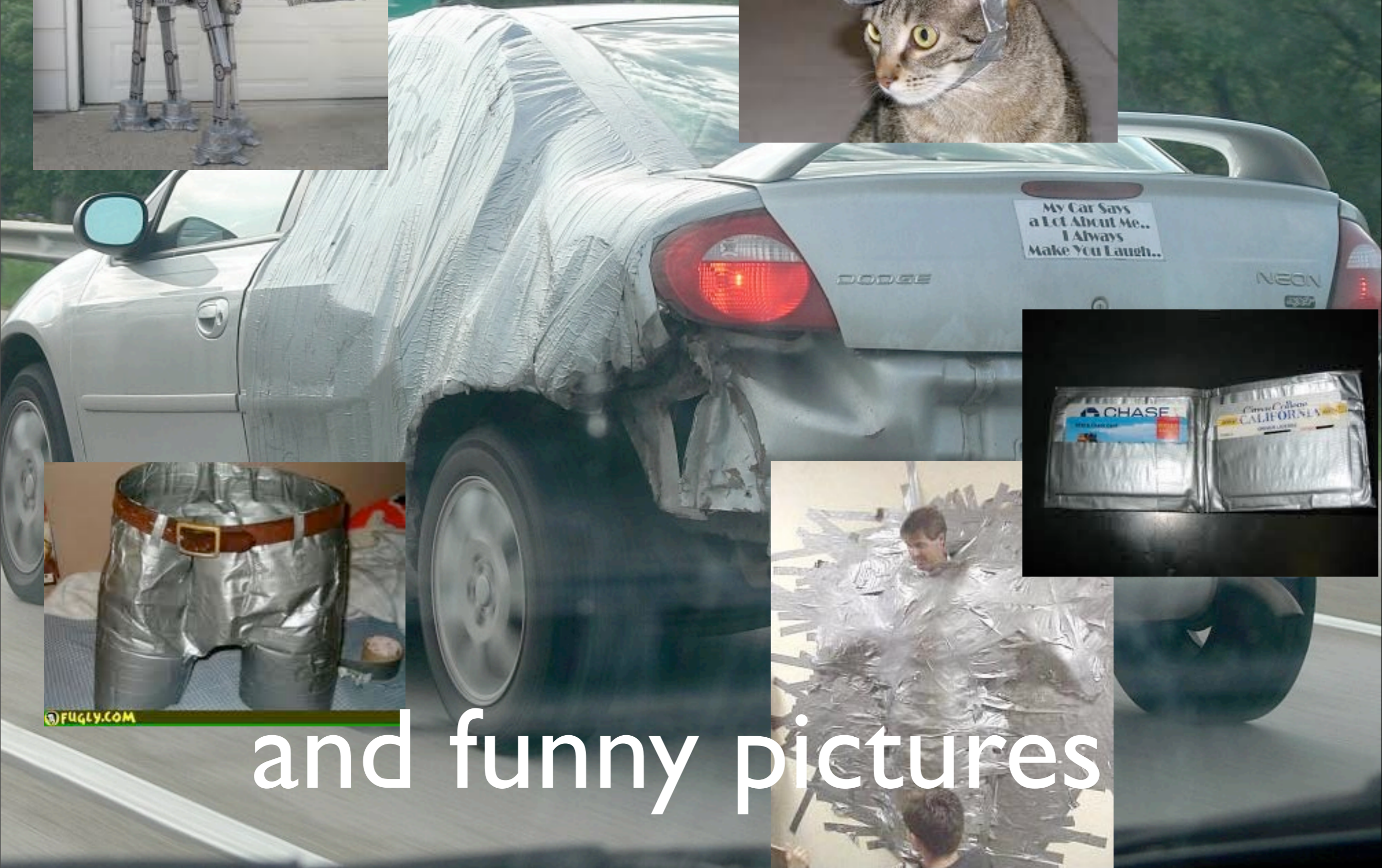
and, Finally

# Restserver (Alpha)

Turn any of Sequels supported databases into a restful service.

jRuby + JDBC + Sinatra + Warbler = Single War File Restservice

<https://github.com/stefanpenner/restserver>



and funny pictures

FUGLY.COM