Redis on Rails

Obie Fernandez

RAILSCONF2012

Redis

- D
- T

APPEND `key value`
Append a value to a key

MGET `key [key ...]`
Get the values of all the given keys

GET `key`
Get the value of a key

PSETEX `key milliseconds value`
Set the value and expiration in milliseconds of a key

GETBIT `key offset`
Returns the bit value at offset in the string value stored at key

SET `key value`
Set the string value of a key

GETRANGE `key start end`
Get a substring of the string stored at a key

SETBIT `key offset value`
Sets or clears the bit at offset in the string value stored at key

GETSET `key value`
Set the string value of a key and return its old value

SETEX `key seconds value`
Set the value and expiration of a key

INCR `key`
Increment the integer value of a key by one

SETNX `key value`
Set the value of a key, only if the key does not exist

Example Code

http://github.com/obie/redis_on_rails

Live Coding

- Adding Redis-based flags and other properties to ActiveRecord objects
- Event tracking with Redis sets
- Graphing relationships between (User) objects with Redis sets
- Time-ordered activity feeds with Redis sorted sets
- Applying security restrictions to display of activity feeds with intersection of Redis sorted sets
- Aggregating group activity feeds with union of Redis sorted sets
- Applying Redis sorted sets to scoring and leaderboard programming
- Integrating Redis with Rspec and Cucumber
- Debugging tactics for when things go wrong or are unclear

```ruby
# config/initializers/redis.rb
$redis = Redis.new(port: 9999)
```

KEYS

Nest

Object-oriented Keys for Redis

github.com/soveran/nest

- Creates a Redis connection by default
- Calls to_s for key representation
- Really simple code / hack it for your needs

```ruby
require "redis"

class Nest < String
  VERSION = "1.1.0"

  METHODS = [:append, :blpop, :brpop, :brpoplpush, :decr, :decrby,
  :del, :exists, :expire, :expireat, :get, :getbit, :getrange, :getset,
  :hdel, :hexists, :hget, :hgetall, :hincrby, :hkeys, :hlen, :hmget,
  :hmset, :hset, :hsetnx, :hvals, :incr, :incrby, :lindex, :linsert,
  :llen, :lpop, :lpush, :lpushx, :lrange, :lrem, :lset, :ltrim, :move,
  :persist, :publish, :rename, :renamenx, :rpop, :rpoplpush, :rpush,
  :rpushx, :sadd, :scard, :sdiff, :sdiffstore, :set, :setbit, :setex,
  :setnx, :setrange, :sinter, :sinterstore, :sismember, :smembers,
  :smove, :sort, :spop, :srandmember, :srem, :strlen, :subscribe,
  :sunion, :sunionstore, :ttl, :type, :unsubscribe, :watch, :zadd,
  :zcard, :zcount, :zincrby, :zinterstore, :zrange, :zrangebyscore,
  :zrank, :zrem, :zremrangebyrank, :zremrangebyscore, :zrevrange,
  :zrevrangebyscore, :zrevrank, :zscore, :zunionstore]

  attr :redis

  def initialize(key, redis = Redis.current)
    super(key.to_s)
    @redis = redis
  end

  def [](key)
    self.class.new("#{self}:#{key}", redis)
  end

  METHODS.each do |meth|
    define_method(meth) do |*args, &block|
      redis.send(meth, self, *args, &block)
    end
  end
end
```

```ruby
# config/initializers/redis.rb
$redis = Redis.new(port: 9999)

# tsk, tsk - use to_param instead of to_s as key
# use our $redis instance as default
Nest.class_eval do
  def initialize(key, redis = $redis)
    super(key.to_param)
    @redis = redis
  end

  def [](key)
    # potential gotchas with slugged models
    self.class.new("#{self}:#{key.to_param}", redis)
  end
end
```

Redis Data Types

- **String**
- List
- Set
- SortedSet
- Hash

APPEND `key value`
Append a value to a key

DECR `key`
Decrement the integer value of a key by one

DECRBY `key decrement`
Decrement the integer value of a key by the given number

GET `key`
Get the value of a key

GETBIT `key offset`
Returns the bit value at offset in the string value stored at key

GETRANGE `key start end`
Get a substring of the string stored at a key

GETSET `key value`
Set the string value of a key and return its old value

INCR `key`
Increment the integer value of a key by one

MGET `key [key ...]`
Get the values of all the given keys

MSET `key value [key value ...]`
Set multiple keys to multiple values

MSETNX `key value [key value ...]`
Set multiple keys to multiple values, only if none of the keys exist

PSETEX `key milliseconds value`
Set the value and expiration in milliseconds of a key

SET `key value`
Set the string value of a key

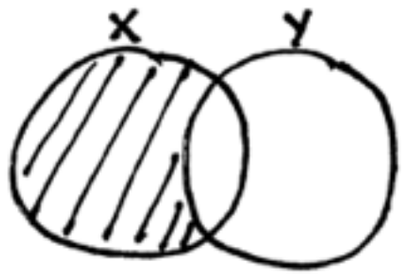
SETBIT `key offset value`
Sets or clears the bit at offset in the string value stored at key

SETEX `key seconds value`
Set the value and expiration of a key

SETNX `key value`
Set the value of a key, only if the key does not exist

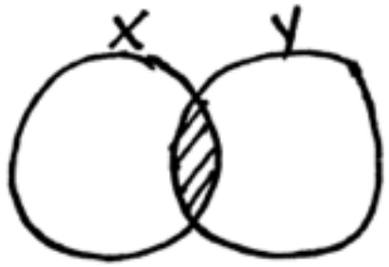
Left side of the equation $X - (X-Y)$

Deal with the parenthesis first



$(X-Y)$

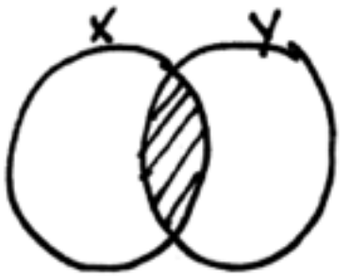
Take all elements in y from X.



$X - (X-Y)$

Take what we found for $(x-y)$ from X, and shade what is left of X.

Right Side of the equation $X \cap Y$



$(X \cap Y)$

The intersection is all of the elements in both x and y.

Two pictures are the same.

$x-(x-y) = x \cap y$

http://upload.wikimedia.org/wikipedia/commons/0/0b/Intersection_of_set_X_and_set_Y_(Venn_Diagrams).jpg

Redis Data Types

- String
- List
- **Set**
- Sorted Set
- Hash

SADD `key member [member ...]`
Add one or more members to a set

SMEMBERS `key`
Get all the members in a set

SCARD `key`
Get the number of members in a set

SMOVE `source destination member`
Move a member from one set to another

SDIFF `key [key ...]`
Subtract multiple sets

SPOP `key`
Remove and return a random member from a set

SDIFFSTORE `destination key [key ...]`
Subtract multiple sets and store the resulting set in a key

SRANDMEMBER `key`
Get a random member from a set

SINTER `key [key ...]`
Intersect multiple sets

SREM `key member [member ...]`
Remove one or more members from a set

SINTERSTORE `destination key [key ...`
Intersect multiple sets and store the resulting set in a key

SUNION `key [key ...]`
Add multiple sets

SISMEMBER `key member`
Determine if a given value is a member of a set

SUNIONSTORE `destination key [key ...`
Add multiple sets and store the resulting set in a key

Redis Data Types

- String
- List
- Set
- **Sorted Set**
- Hash

ZADD `key score member [score] [member]`
Add one or more members to a sorted set, or update its score if it already exists

ZREM `key member [member ...]`
Remove one or more members from a sorted set

ZCARD `key`
Get the number of members in a sorted set

ZREMRANGEBYRANK `key start stop`
Remove all members in a sorted set within the given indexes

ZCOUNT `key min max`
Count the members in a sorted set with scores within the given values

ZREMRANGEBYSCORE `key min max`
Remove all members in a sorted set within the given scores

ZINCRBY `key increment member`
Increment the score of a member in a sorted set

ZREVRANGE `key start stop [WITHSCORES]`
Return a range of members in a sorted set, by index, with scores ordered from high to low

ZINTERSTORE `destination numkeys key [key ...`
Intersect multiple sorted sets and store the resulting sorted set in a new key

ZREVRANGEBYSCORE `key max min [WITHSCORES...`
Return a range of members in a sorted set, by score, with scores ordered from high to low

ZRANGE `key start stop [WITHSCORES]`
Return a range of members in a sorted set, by index

ZREVRANK `key member`
Determine the index of a member in a sorted set, with scores ordered from high to low

ZRANGEBYSCORE `key min max [WITHSCORES] [L...`
Return a range of members in a sorted set, by score

ZSCORE `key member`
Get the score associated with the given member in a sorted set

ZRANK `key member`
Determine the index of a member in a sorted set

ZUNIONSTORE `destination numkeys key [key ...`
Add multiple sorted sets and store the resulting sorted set in a new key

Redis Data Types

- String
- List
- Set
- Sorted Set
- **Hash**

HDEL `key field [field ...]`
Delete one or more hash fields

HLEN `key`
Get the number of fields in a hash

HEXISTS `key field`
Determine if a hash field exists

HMGET `key field [field ...]`
Get the values of all the given hash fields

HGET `key field`
Get the value of a hash field

HMSET `key field value [field value ...`
Set multiple hash fields to multiple values

HGETALL `key`
Get all the fields and values in a hash

HSET `key field value`
Set the string value of a hash field

HINCRBY `key field increment`
Increment the integer value of a hash field by the given number

HSETNX `key field value`
Set the value of a hash field, only if the field does not exist

HINCRBYFLOAT `key field increment`
Increment the float value of a hash field by the given amount

HVALS `key`
Get all the values in a hash

HKEYS `key`
Get all the fields in a hash

RedisProps

Easy annotation of ActiveRecord objects. Extracted from DueProps

http://github.com/obie/redis_props

example: facebook_authentication.rb

RedisObjects

Map Redis types directly to Ruby objects

github.com/nateware/redis-objects

http://nateware.com/2010/02/18/an-atomic-rant/