Redis - justificativa de uso

O Redis é um SGBD não relacional do tipo chave-valor que guarda dados diretamente na memória. Pode ser usado tanto como banco de dados como cache. É usado para guardar hashes, strings e listas (principalmente - pois permite também alguns outros usos).

No nosso sistema, a intenção de uso do Redis é guardar um dicionário que indica quais aulas foram mais acessadas recentemente. Para isso, pretendemos, sempre que uma aula for acessada, incrementar um contador que guardaremos usando o Redis. Esse contador nada mais será do que uma chave ('id' da aula) apontando para um valor (número de vezes em que ela foi assistida recentemente).

O motivo principal de usarmos esse SGBD e não outro é o fato de que podemos fazer operações e recuperações de forma bastante rápida (pois estamos lidando diretamente com a memória). Além disso, o tipo de dado que queremos guardar é simples, e executar comandos num SGBDR para administrar isso seria mais denso (exigiria mais comandos, deixando um código menos legível e mais 'poluído'), além de mais complicado (usando o Redis - se verá mais adiante - bastam poucas 'palavras' para inserir um novo contador e atualizá-lo).

Eventualmente, outras operações podem vir a fazer sentido usando esse SGBD e suas operações - no entanto, simplesmente sua versatilidade, velocidade e facilidade para administrar os contadores (e, com isso, um 'ranking' de aulas) foram suficiente para que optássemos por ele.

O método de administração desses contadores ainda está sendo estudado - não basta somente incrementá-los; há também que decrementá-los periodicamente de acordo com algum critério ou algoritmo, de forma que a classificação faça sentido.

De posse desse dicionário, teremos a possibilidade de sugerir aos usuários as aulas que estão populares no momento - isso seria bastante interessante, por exemplo, para um usuário que quer participar de alguma para testar o sistema, ou que queira aprender algo novo mas não tem nada específico em mente.

Redis - instalação

Utilizamos, como referência, o próprio site do Redis (http://redis.io/), nele foi possível solucionar todas as nossas duvidas para a sua instalação e queries. Baixar o redis é bastante simples: basta rodar os comandos listados abaixo:

```
$ wget http://download.redis.io/releases/redis-3.0.5.tar.gz
$ tar xzf redis-3.0.5.tar.gz
$ cd redis-3.0.5
$ make
```

no bash (ou terminal Linux preferido). Linha a linha, faz o seguinte:

1) Baixa o pacote comprimido (.tar.gz);
2) O descompacta;
3) Entra no diretório em que se encontra o pacote extraído e
4) Roda as rotinas e compilação e instalação (automatizadas pelo Makefile).

Uma vez rodado o comando 'make', é simples executar o servidor do SGBD:

```
$ src/redis-server
```

Com o comando acima, uma interface do SGBD será aberta (um interpretador de linha de comando) e se poderá começar a dar os comandos. Os comandos descritos mais em adiante neste documento são comandos que devem ser executados **na linha de comando do servidor Redis**. Quando integrarmos o SGBD com nossa aplicação, será necessário encontrar a interface adequada. Por ora, se utilizará e fará referência sempre ao servidor padrão.

Redis - inserção

Para inserir um dado no Redis, se utiliza o método **SET**, segundo a seguinte sintaxe:

```
redis> SET mykey "Hello"
OK
```

Ou seja: **SET <chave> <valor>.** No exemplo dado, foi usada uma string - no caso do nosso projeto, usaremos números inteiros. Por exemplo:

```
SET view_counter_for_class_3124 0
```

Que significa inicializar o contador de visualizações para a aula de número 3124. É importante ressalvar que as chaves nesse tipo de armazenamento são muito importantes, e deve-se preferencialmente criar uma convenção fácil de memorizar e consistente, para que se possa manipular e acessar os dados requeridos.

Redis - recuperação

Recuperar um dado nesse SGBD é tão simples como inseri-lo: basta usar o comando **GET.**

```
redis> GET mykey
"Hello"
```

Nesse exemplo, o comando retorna "Hello", de acordo com o comando SET executado no exemplo anterior.

Incrementar e decrementar

Duas operações que utilizaremos muito - mais do que qualquer outra - serão a de incrementar/decrementar os contadores. Pelo comportamento peculiar do Redis, decidimos descrever também as operações que usaremos com mais frequência.

Para incrementar um valor (dada sua chave):

```
redis> SET mykey "10"
OK
redis> INCR mykey
(integer) 11
```

Note-se que a atribuição do valor para a chave ‘mickey’ é feita como string (isso faz parte do comportamento padrão para ‘atribuição’ no Redis). Quando o incrementamos, ele retorna o valor (incrementado) acusando-o como inteiro. Porém, ao usar novamente o getter para a dada chave:

```
redis> GET mykey
"11"
```

Obtemos novamente uma ‘string’. Isso acontece porque o Redis não possui um tipo inteiro: inicialmente ele armazena inteiros como strings. Se o comando **INCR** for executado sobre uma chave que guarda uma string que não pode ser convertida em inteiro, um erro é retornado (o mesmo vale para outros tipos de dados incompatíveis).

Para decrementar, o comando é análogo:

```
redis> SET mykey "10"
OK
redis> DECR mykey
(integer) 9
redis> SET mykey "234293482390480948029348230948"
OK
redis> DECR mykey
ERR value is not an integer or out of range
```