

# Re: Best database for implementing a cache

---

*Source:* <http://coding.derkeiler.com/Archive/Java/comp.lang.java.programmer/2007-03/msg03528.html>

---

- *From:* "vj" <[mr.vaibhavjain@xxxxxxxxxx](mailto:mr.vaibhavjain@xxxxxxxxxx)>
  - *Date:* 31 Mar 2007 00:36:39 -0700
- 

On Mar 31, 6:11 am, Arne Vajhøj <[a...@xxxxxxxxxx](mailto:a...@xxxxxxxxxx)> wrote:

vj wrote:

Why not just write then to the disk directly? Most File Systems can be treated as a form a database, and it seems like thats kind of what you want anyway.

Yes i am writing the image files directly to the filesystem as it is. However i need to limit the size of the cache so that it may not grow arbitrarily large. that would require me to maintain access count metric for evicting existing stored images. Since filesystem itself would not let me associate this info with the image I thought of a utilizing a database that would store the access count as well as all the available images in the cache.

What about keeping files on disk, having meta data in memory and simply rebuilding meta data at startup (list all files on disk and start with zero counters) ?

Arne

Certainly thats a good idea. How aboute mainting a hashtable and a priority queue. The hashtable will store all the meta data associated with a file and the priority queue will implement file aviction policy. as soon as a a file is hit i will increase the hit count. On a cache miss i will fetch the image , create an entry in the hash table and push an entry in the queue. if the queue size if creater than a certain treshhold i will pop the element from its head (with the lowest access count) delete its corosponding file / hashtable entry.

This seems to be a good idea but i have a small doubt. Does propirity queue dynamicaly orders elements. I mean since the priority queue is

## Re: Best database for implementing a cache

implemented as a heap in java hence element ordering only takes place when elements are added. This will create problems as because if i update access count in hashtable then they wont be reflected in the structure of the queue. Hence when i pop element from its head for eviction it might not be the one with minimum access count.

What do you think, Any other data structure that we can use for fast lookup of elements

.