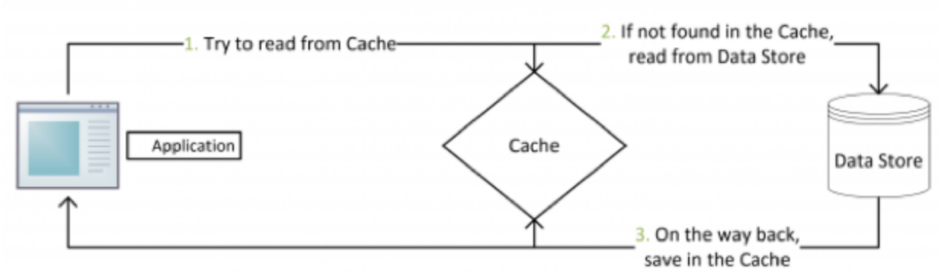
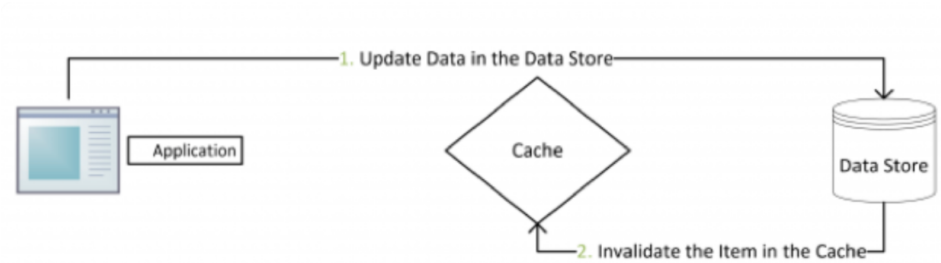
三种缓存设计模式

Cache Aside 更新模式





- 失效：应用程序先从 Cache 取数据, 如果没有得到, 再从数据库中取, 成功后, 再放到缓存中
- **命中**：应用程序从 Cache 中取数据, 取到后返回
- **更新**：先把数据存到数据库中, 成功后, 再让缓存失效
- 为什么是让缓存失效而不是更新缓存呢?
  - 主要是防止 两个并发写操作导致脏数据
  - 存在两个并发写，w1, w2, 可能会w1 先写数据库， 然后发送写缓存操作c1， 然后w2 写， 发送更新缓存操作c2， 可能的顺序是 w1, w2,c2,c1 然后有脏数据。

Read/Write through 更新模式

- 由 Cache 自己更新数据库, 应用程序只需要认为后端是一个单一的存储
- 不同的是, Cache Aside 需要同时维护缓存和数据库
- Read Through：如果缓存失效, 缓存服务自己在查询时更新缓存
- Write Through：更新数据时, 如果命中了缓存, 由Cache自己更新数据库

Write-Back 更新模式

- 就是 Linux 文件系统中的 Page Cache 算法
- 在更新数据时, 只更新缓存, 不更新数据库, 而我们的缓存会批量的异步的更新数据库.
- 这个设计的好处是让数据的I/O操作飞快, 因为直接操作内存; 因为异步, 还可以合并对同一数据的多次操作, 所以性能的提高是相当可观的
- 但其带来的问题是，数据不是强一致性的，而且可能会丢失（我们知道 Unix/Linux 非正常关机会导致数据丢失，就是因为这个事）