```
┌──────────────────── MODULE cacheinvalidationv3 ────────────────────┐
EXTENDS Naturals

CONSTANTS
    KEYS


VARIABLES
    database,
    cache,
    cacheFillStates,
    invalidationQueue,
    counter  Used to prevent repeated states for liveness conditions


INSTANCE cacherequirements

vars ≜ ⟨database, cache, cacheFillStates, invalidationQueue, counter⟩

InvalidationMessage ≜ [key : KEYS, version : DataVersion]

CacheFillState ≜ [state : {"inactive", "started", "respondedto"}, version : DataVersion]

CacheValue ≜ CacheMiss ∪ CacheHit

TypeOk ≜
    ∧  database ∈ [KEYS → DataVersion]
    ∧  cache ∈ [KEYS → CacheValue]
    ∧  cacheFillStates ∈ [KEYS → CacheFillState]
    ∧  invalidationQueue ∈ SUBSET InvalidationMessage
    ∧  counter ∈ Nat

Init ≜
    ∧ database = [k ∈ KEYS ↦ 0]
    ∧ cache = [k ∈ KEYS ↦ [type  ↦ "miss"]]
    ∧ cacheFillStates = [k ∈ KEYS ↦ [
                                state ↦ "inactive",
                                version ↦ 0]
                            ]
    ∧ invalidationQueue = {}
    ∧ counter = 0


DatabaseUpdate(k) ≜
    LET updatedVersion ≜ database[k] + 1IN
    ∧ database′ = [database EXCEPT
                        ![k] = updatedVersion]
    ∧ invalidationQueue′ = invalidationQueue ∪
```

```
                              $\{[key \mapsto k,\ version \mapsto updatedVersion]\}$
    $\land$ UNCHANGED $\langle cache,\ cacheFillStates,\ counter\rangle$

$CacheStartReadThroughFill(k) \triangleq$
    $\land\ cache[k] \in CacheMiss$
    $\land\ cacheFillStates[k].state =$ "inactive"
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT $![k].state =$ "started"$]$
    $\land$ UNCHANGED $\langle database,\ cache,\ invalidationQueue,\ counter\rangle$

$DatabaseRespondToCacheFill(k) \triangleq$
    $\land\ cacheFillStates[k].state =$ "started"
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT
                              $![k].state =$ "respondedto",
                              $![k].version = database[k]$
                             $]$
    $\land$ UNCHANGED $\langle database,\ cache,\ invalidationQueue,\ counter\rangle$

$CacheFailFill(k) \triangleq$
    $\land\ cacheFillStates[k].state =$ "respondedto"
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT
                              $![k].state =$ "inactive",
                              $![k].version = 0$
                             $]$
    $\land$ UNCHANGED $\langle database,\ cache,\ invalidationQueue,\ counter\rangle$

$CacheCompleteFill(k) \triangleq$
    $\land\ cacheFillStates[k].state =$ "respondedto"
    $\land\ \lor\ cache[k] \in CacheMiss$
      $\lor\ \land\ cache[k] \notin CacheMiss$
         $\land\ cache[k].version < cacheFillStates[k].version$
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT   Reset to 0
                              $![k].state =$ "inactive",
                              $![k].version = 0$
                             $]$
    $\land\ cache' = [cache$ EXCEPT
                        $![k] = [$
                           $type \mapsto$ "hit",
                           $version \mapsto cacheFillStates[k].version$
                        $]$
                 $]$
    $\land$ UNCHANGED $\langle database,\ invalidationQueue,\ counter\rangle$

$CacheIgnoreFill(k) \triangleq$
    $\land\ cacheFillStates[k].state =$ "respondedto"
    $\land\ \land\ cache[k] \in CacheHit$
        $\land\ cache[k].version \geq cacheFillStates[k].version$
    $\land\ cacheFillStates' = [cacheFillStates$ EXCEPT
                                  $![k].state =$ "inactive",
                                  $![k].version = 0$
                                 $]$
    $\land\ counter' = counter + 1$
    $\land$ UNCHANGED $\langle cache,\ database,\ invalidationQueue \rangle$

$CacheHandleInvalidationMessage\ \triangleq$
    $\land\ \exists\ message \in invalidationQueue :$
        $\land\ \lor\ \land\ cache[message.key] \in CacheHit$
                Message needs to be newer than the cache
                $\land\ cache[message.key].version < message.version$
              Or not in the cache, but with a pending fill request
            $\lor\ \land\ cache[message.key] \in CacheMiss$
                $\land\ cacheFillStates[message.key].state \neq$ "inactive"
        Update item in cache
        $\land\ cache' = [cache$ EXCEPT
                          $![message.key] = [$
                              $type \mapsto$ "hit",
                               Update to version in invalidation message
                              $version \mapsto message.version$
                          $]]$
        Remove message from queue because handled
        $\land\ invalidationQueue' = invalidationQueue \setminus \{message\}$

    $\land$ UNCHANGED $\langle cacheFillStates,\ database,\ counter \rangle$

$CacheIgnoreInvalidationMessage\ \triangleq$
    $\land\ \exists\ message \in invalidationQueue :$  Dequeue invalidation queue in any order
        Ignore invalidation messages for messages not in cache
        $\land\ \lor\ \land\ cache[message.key] \in CacheMiss$
                and a fill is not occurring
                $\land\ cacheFillStates[message.key].state =$ "inactive"
              Or when the cache already has the same or larger version
            $\lor\ \land\ cache[message.key] \notin CacheMiss$
                $\land\ cache[message.key].version \geq message.version$
        Remove message from queue to ignore
        $\land\ invalidationQueue' = invalidationQueue \setminus \{message\}$
    $\land\ counter' = counter + 1$
    $\land$ UNCHANGED $\langle cacheFillStates,\ database,\ cache \rangle$

$CacheEvict(k)\ \triangleq$
    $\land\ cache[k] \in CacheHit$
    A key with a pending request will not be evicted
    $\land\ \mathit{cacheFillStates}[k].\mathit{state} = \text{“inactive”}$
    $\land\ \mathit{cache}' = [\mathit{cache}\ \textsc{except}\ ![k] = [\mathit{type} \mapsto \text{“miss”}]]$
    $\land\ \mathit{counter}' = \mathit{counter} + 1$
    $\land\ \textsc{unchanged}\ \langle \mathit{database},\ \mathit{cacheFillStates},$
                      $\mathit{invalidationQueue} \rangle$

$\mathit{CacheFairness}\ \triangleq$
  $\exists\, k \in \mathit{KEYS} :$
      $\lor\ \mathit{CacheStartReadThroughFill}(k)$
      $\lor\ \mathit{DatabaseRespondToCacheFill}(k)$
      $\lor\ \mathit{CacheCompleteFill}(k)$
      $\lor\ \mathit{CacheIgnoreFill}(k)$
      $\lor\ \mathit{CacheHandleInvalidationMessage}$
      $\lor\ \mathit{CacheIgnoreInvalidationMessage}$
```

## Specification

```
$\mathit{Next}\ \triangleq$
  $\exists\, k \in \mathit{KEYS} :$
      Database states
      $\lor\ \mathit{DatabaseUpdate}(k)$
      Cache states
      $\lor\ \mathit{CacheStartReadThroughFill}(k)$
      $\lor\ \mathit{DatabaseRespondToCacheFill}(k)$
      $\lor\ \mathit{CacheCompleteFill}(k)$
      $\lor\ \mathit{CacheIgnoreFill}(k)$
      $\lor\ \mathit{CacheHandleInvalidationMessage}$
      $\lor\ \mathit{CacheIgnoreInvalidationMessage}$
      $\lor\ \mathit{CacheEvict}(k)$

$\mathit{Spec}\ \triangleq\ \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}} \land \mathrm{WF}_{\mathit{vars}}(\mathit{CacheFairness})$
```

* Modification History
* Last modified *Wed Jun* 15 13:08:32 *MST* 2022 by *elliotswart*
* Created *Tue Jun* 14 20:36:02 *MST* 2022 by *elliotswart*