MODULE *cacherequirements*

EXTENDS *Naturals*

CONSTANTS
*KEYS* The full set of keys in the database

VARIABLES
*database*, *database*[*key*] = *DataVersion*
*cache* *cache*[*key*] = *CacheValue*

The maximum number of versions a key can have in this model
$MaxVersions \triangleq 4$

Data versions are scoped to an individual key
$DataVersion \triangleq Nat$

Represents the absence of a value in a cache
$CacheMiss \triangleq [type : \{\text{“miss”}\}]$

Represents the presence of a value in a cache, as well as the value
$CacheHit \triangleq [type : \{\text{“hit”}\}, version : DataVersion]$

$DatabaseAndCacheConsistent \triangleq$
$\forall k \in KEYS :$
If the key is in cache
$\lor \land cache[k] \in CacheHit$
It should be the same version as the database
$\land cache[k].version = database[k]$
A cache miss is also okay. A cache won't hold everything
$\lor cache[k] \in CacheMiss$

This means that at some point, the database and cache are consistent.
It is important to note that this is not eventual consistency.
This only says it needs to be eventually consistent once.
$EventuallyDatabaseAndCacheConsistent \triangleq \Diamond DatabaseAndCacheConsistent$

The cache must be always eventually consistent.
$AlwaysEventuallyDatabaseAndCacheConsistent \triangleq$
$\Box EventuallyDatabaseAndCacheConsistent$

Used as a state constraint to prevent unbounded testing
with infinite versions.
$DatabaseRecordsDoNotExceedMaxVersion \triangleq$
$\forall k \in KEYS :$
$database[k] < MaxVersions$

\ * Modification History
\ * Last modified *Tue Jun* 14 22:44:55 *MST* 2022 by *elliotswart*
\ * Created *Tue Jun* 14 21:36:26 *MST* 2022 by *elliotswart*