

```

1  |----- MODULE basicakgi -----|
   | a simplified specification for uniqorn. (1) a time oracle is used for versioning
   | (2) we omit update operations, which have the same effect as create operations,
   | (3) each record has only and exactly one alternate key. (4) a background garbage cleanup process
   | is used to delete garbage index records. (5) when a create needs to reuse a garbage index record,
   | it will not delete it, i.e. no mandatory garbage cleanup is used, only the background garbage
   | cleanup process will delete garbage index records.
11 EXTENDS TLC, Integers, Sequences, FiniteSets, Bags

13 | phases for create (update operations have the same effect, so are omitted)
14 CONSTANTS CREATE_INIT_DATA_RECORD
15 CONSTANTS CREATE_PERSIST_INDEX_RECORD
16 CONSTANTS CREATE_PERSIST_DATA_RECORD

18 | phases for cleanup
19 CONSTANTS CLEANUP_VALIDATE
20 CONSTANTS CLEANUP_CHANGE_LOCK
21 CONSTANTS CLEANUP_DELETE_GARBAGE

23 | delete has only one phase, so we ignore it

25 | set of integer keys for primary keys and alternate keys
26 CONSTANTS PKS, AKS

28 | a non-zero integer
29 CONSTANTS VAL

31 | data or index records in data store partitions or index store partitions
32 VARIABLES persistedDataRecords, persistedIndexRecords

34 | separate queues for all create/cleanup operations, delete has only one phase,
35 | no need a queue for it.
36 | an operation can enqueue and dequeue as it progress through its various phases.
37 | no operations, once
38 | enqueued, will be dequeued, in order to emulate duplicated operations
39 VARIABLES inprogressCreates, inprogressCleanups

41 | we use global monotonic timestamp for the basic case
42 VARIABLES timestamp

44 | ***** data store accesses start here *****
45 IsDummy(pk)  $\triangleq$  IF  $\wedge pk \in \text{DOMAIN } \textit{persistedDataRecords}$ 
46  $\wedge \textit{persistedDataRecords}[pk].ak = 0$ 
47  $\wedge \textit{persistedDataRecords}[pk].val = 0$ 
48 THEN TRUE
49 ELSE FALSE

51 IsStale(pk, ts)  $\triangleq$  IF  $\wedge pk \in \text{DOMAIN } \textit{persistedDataRecords}$ 
52  $\wedge \textit{persistedDataRecords}[pk].ts > ts$ 

```

```

53             THEN TRUE
54             ELSE FALSE

56 isLockHeld(pk, ts)  $\triangleq$  IF  $\wedge$  pk  $\in$  DOMAIN persistedDataRecords
57              $\wedge$  persistedDataRecords[pk].ts = ts
58             THEN TRUE
59             ELSE FALSE

61 DataStoreDelete(pk)  $\triangleq$ 
62      $\wedge$  pk  $\in$  DOMAIN persistedDataRecords
63      $\wedge$  persistedDataRecords' = [key  $\in$  (DOMAIN persistedDataRecords \
64         {pk})  $\mapsto$  persistedDataRecords[key]]
65      $\wedge$  UNCHANGED (persistedIndexRecords, inprogressCreates, inprogressCleanups)

67 DataStoreInitLock(pk, ak, ts)  $\triangleq$ 
68      $\vee$   $\wedge$  pk  $\notin$  DOMAIN persistedDataRecords
69      $\wedge$  persistedDataRecords' = persistedDataRecords @@ (pk :> [pk  $\mapsto$  pk, ts  $\mapsto$  ts, ak  $\mapsto$  0, val  $\mapsto$  0])
70      $\wedge$  inprogressCreates' = inprogressCreates  $\cup$  {[phase  $\mapsto$  CREATE_PERSIST_INDEX_RECORD,
71         pk  $\mapsto$  pk, ak  $\mapsto$  ak, ts  $\mapsto$  ts]}
72      $\vee$   $\wedge$  IsDummy(pk)
73      $\wedge$   $\neg$  IsStale(pk, ts)
74      $\wedge$  persistedDataRecords' = [persistedDataRecords EXCEPT ![pk].ts = ts]
75      $\wedge$  inprogressCreates' = inprogressCreates  $\cup$  {[phase  $\mapsto$  CREATE_PERSIST_INDEX_RECORD,
76         pk  $\mapsto$  pk, ak  $\mapsto$  ak, ts  $\mapsto$  ts]}
77      $\vee$  UNCHANGED (persistedDataRecords, inprogressCreates)

79 DataStoreUpdateOptimistically(pk, ak, ts)  $\triangleq$ 
80      $\vee$   $\wedge$  isLockHeld(pk, ts)
81      $\wedge$  persistedDataRecords' = [persistedDataRecords EXCEPT ![pk].ts = @ + 1,
82         ![pk].ak = ak, ![pk].val = VAL]
83      $\vee$  UNCHANGED persistedDataRecords

85 DataStoreValidate(pk, ak, ts)  $\triangleq$ 
86     IF pk  $\in$  DOMAIN persistedDataRecords  $\wedge$  persistedDataRecords[pk].ak = ak THEN
87         UNCHANGED inprogressCleanups
88     ELSE
89         inprogressCleanups' = inprogressCleanups  $\cup$  {[phase  $\mapsto$  CLEANUP_CHANGE_LOCK, pk  $\mapsto$  pk,
90             ak  $\mapsto$  ak, ts  $\mapsto$  ts]}

92 DataStoreChangeLock(pk, ak, ts)  $\triangleq$ 
93     IF pk  $\in$  DOMAIN persistedDataRecords THEN
94         IF ak  $\neq$  persistedDataRecords[pk].ak THEN
95              $\wedge$  IF persistedDataRecords[pk].val = 0 THEN
96                 persistedDataRecords' = [key  $\in$  (DOMAIN persistedDataRecords \
97                     {pk})  $\mapsto$  persistedDataRecords[key]]
98             ELSE
99                 persistedDataRecords' = [persistedDataRecords EXCEPT ![pk].ts = @ + 1]

```

```

100       $\wedge \text{inprogressCleanups}' = \text{inprogressCleanups} \cup \{[phase \mapsto \text{CLEANUP\_CHANGE\_LOCK},$ 
101       $pk \mapsto pk, ak \mapsto ak, ts \mapsto ts]\}$ 
102      ELSE UNCHANGED  $\langle \text{persistedDataRecords}, \text{inprogressCleanups} \rangle$ 
103  ELSE
104       $\wedge \text{inprogressCleanups}' = \{\text{inprogressCleanups}\} \cup \{[phase \mapsto \text{CLEANUP\_DELETE\_GARBAGE},$ 
105       $pk \mapsto pk, ak \mapsto ak, ts \mapsto ts]\}$ 
106       $\wedge$  UNCHANGED  $\text{persistedDataRecords}$ 

108  data store partitioning/routing policies do not affect the correctness, so we ignore them
109  ***** data store accesses start here *****

111  ***** index store accesses start here *****
112  index store has only two accesses methods: insert and delete. Update and replace accesses can
113  be derived from these two acceses.
114   $\text{IndexStoreDirectlyInsert}(ak, pk, ts) \triangleq$ 
115       $\vee \wedge ak \notin \text{DOMAIN } \text{persistedIndexRecords}$ 
116       $\wedge \text{persistedIndexRecords}' = \text{persistedIndexRecords} \quad @@(ak \mapsto [ak \mapsto ak, pk \mapsto pk,$ 
117       $ts \mapsto ts])$ 
118       $\wedge \text{inprogressCreates}' = \text{inprogressCreates} \cup \{[phase \mapsto \text{CREATE\_PERSIST\_DATA\_RECORD},$ 
119       $pk \mapsto pk, ak \mapsto ak, ts \mapsto ts]\}$ 
120       $\vee$  UNCHANGED  $\langle \text{persistedIndexRecords}, \text{inprogressCreates} \rangle$ 

122   $\text{IndexStoreDeleteOptimistically}(ak, pk, ts) \triangleq$ 
123  IF  $\wedge ak \in \text{DOMAIN } \text{persistedIndexRecords}$ 
124       $\wedge \text{persistedIndexRecords}[ak].pk = pk$ 
125       $\wedge \text{persistedIndexRecords}[ak].ts = ts$ 
126  THEN
127       $\text{persistedIndexRecords}' = [key \in (\text{DOMAIN } \text{persistedIndexRecords} \setminus$ 
128       $\{ak\}) \mapsto \text{persistedIndexRecords}[key]]$ 
129  ELSE UNCHANGED  $\text{persistedIndexRecords}$ 
130  ***** index store accesses end here *****

134  make a create operation go through its phases
135   $\text{RunCreate}(\text{createOp}) \triangleq$ 
136      LET  $phase \triangleq \text{createOp}.phase$ 
137       $pk \triangleq \text{createOp}.pk$ 
138       $ak \triangleq \text{createOp}.ak$ 
139       $ts \triangleq \text{createOp}.ts$ 
140  IN  $\vee \wedge phase = \text{CREATE\_INIT\_DATA\_RECORD}$ 
141       $\wedge \text{DataStoreInitLock}(pk, ak, ts)$ 
142       $\wedge$  UNCHANGED  $\langle \text{persistedIndexRecords}, \text{inprogressCleanups} \rangle$ 
143       $\vee \wedge phase = \text{CREATE\_PERSIST\_INDEX\_RECORD}$ 
144       $\wedge \text{IndexStoreDirectlyInsert}(ak, pk, ts)$ 
145       $\wedge$  UNCHANGED  $\langle \text{persistedDataRecords}, \text{inprogressCleanups} \rangle$ 
146       $\vee \wedge phase = \text{CREATE\_PERSIST\_DATA\_RECORD}$ 

```

147 $\wedge \text{DataStoreUpdateOptimistically}(pk, ak, ts)$
 148 $\wedge \text{UNCHANGED } \langle \text{persistedIndexRecords}, \text{inprogressCleanups}, \text{inprogressCreates} \rangle$
 150 **issue a create operation**
 151 $\text{Create}(pk, ak) \triangleq$
 152 $\wedge \text{inprogressCreates}' = \text{inprogressCreates} \cup \{ [phase \mapsto \text{CREATE_INIT_DATA_RECORD},$
 153 $pk \mapsto pk, ak \mapsto ak, ts \mapsto \text{timestamp}] \}$
 154 $\wedge \text{UNCHANGED } \langle \text{persistedDataRecords}, \text{persistedIndexRecords}, \text{inprogressCleanups} \rangle$
 156 **issue a cleanup operation**
 157 $\text{Cleanup}(ak, pk, ts) \triangleq$
 158 $\wedge \text{inprogressCleanups}' = \text{inprogressCleanups} \cup \{ [phase \mapsto \text{CLEANUP_VALIDATE}, ak \mapsto ak,$
 159 $pk \mapsto pk, ts \mapsto ts] \}$
 160 $\wedge \text{UNCHANGED } \langle \text{persistedDataRecords}, \text{persistedIndexRecords}, \text{inprogressCreates} \rangle$
 162 **make a garbage cleanup operation go through its phases**
 163 $\text{RunCleanup}(\text{cleanupOp}) \triangleq$
 164 $\text{LET } phase \triangleq \text{cleanupOp.phase}$
 165 $pk \triangleq \text{cleanupOp.pk}$
 166 $ak \triangleq \text{cleanupOp.ak}$
 167 $ts \triangleq \text{cleanupOp.ts}$
 168 $\text{IN } \vee \wedge phase = \text{CLEANUP_VALIDATE}$
 169 $\wedge \text{DataStoreValidate}(pk, ak, ts)$
 170 $\wedge \text{UNCHANGED } \langle \text{persistedDataRecords}, \text{persistedIndexRecords}, \text{inprogressCreates} \rangle$
 171 $\vee \wedge phase = \text{CLEANUP_CHANGE_LOCK}$
 172 $\wedge \text{DataStoreChangeLock}(pk, ak, ts)$
 173 $\wedge \text{UNCHANGED } \langle \text{persistedIndexRecords}, \text{inprogressCreates} \rangle$
 174 $\vee \wedge phase = \text{CLEANUP_DELETE_GARBAGE}$
 175 $\wedge \text{IndexStoreDeleteOptimistically}(ak, pk, ts)$
 176 $\wedge \text{UNCHANGED } \langle \text{persistedDataRecords}, \text{inprogressCreates}, \text{inprogressCleanups} \rangle$
 178 **all aks and pks are initialized in each partition**
 179 $\text{Init} \triangleq \wedge \text{persistedDataRecords} = [pk \in \{\} \mapsto \{\}]$
 180 $\wedge \text{persistedIndexRecords} = [ak \in \{\} \mapsto \{\}]$
 181 $\wedge \text{inprogressCreates} = \{\}$
 182 $\wedge \text{inprogressCleanups} = \{\}$
 183 $\wedge \text{timestamp} = 0$
 185 $\text{Next} \triangleq \wedge \vee \exists pk \in \text{PKS} : \text{DataStoreDelete}(pk)$
 186 $\vee \exists pk \in \text{PKS}, ak \in \text{AKS} : \text{Create}(pk, ak)$
 187 $\vee \exists ak \in \text{DOMAIN } \text{persistedIndexRecords} : \text{Cleanup}(ak,$
 188 $\text{persistedIndexRecords}[ak].pk, \text{persistedIndexRecords}[ak].ts)$
 189 $\vee \exists \text{createOp} \in \text{inprogressCreates} : \text{RunCreate}(\text{createOp})$
 190 $\vee \exists \text{cleanupOp} \in \text{inprogressCleanups} : \text{RunCleanup}(\text{cleanupOp})$
 191 $\wedge \text{timestamp}' = \text{timestamp} + 1$
 193 $\text{Spec} \triangleq \text{Init} \wedge \Box[\text{Next}] \langle \text{persistedDataRecords}, \text{persistedIndexRecords}, \text{inprogressCreates},$

```

194                                     inprogressCleanups, timestamp⟩
196  no missing index record invariant
197  NoMissing  $\triangleq \forall pk \in \text{DOMAIN } persistedDataRecords :$ 
198      IF persistedDataRecords[pk].ak  $\neq 0$  THEN
199           $\exists ak \in \text{DOMAIN } persistedIndexRecords :$ 
200               $\wedge persistedIndexRecords[ak].pk = pk$ 
201               $\wedge persistedDataRecords[pk].ak = ak$ 
202          ELSE TRUE
204  THEOREM Spec  $\Rightarrow$  NoMissing
205  ┌
    \ * Modification History
    \ * Last modified Wed Mar 07 17:22:58 PST 2018 by jyi
    \ * Created Thu Feb 01 13:21:10 PST 2018 by jyi

```