```
──────────────────── MODULE AggCount ────────────────────
EXTENDS TLC, Integers, FiniteSets

CONSTANTS Dataset, Storage, nil

VARIABLES replicas, pending_counters

vars ≜ ⟨replicas, pending_counters⟩

min_repl_id ≜ 21
max_repl_id ≜ 25

ReplicaID ≜ min_repl_id .. max_repl_id

Status ≜ {"pending", "written"}

ReplicaInfo ≜ [ds : Dataset, status : Status, storage : Storage, agg : BOOLEAN ]

Replica ≜ [ReplicaID → ReplicaInfo ∪ {nil}]

PendingKey ≜ Dataset × Storage

PendingInfo ≜ [count : 0 .. 100, need_update : BOOLEAN , version : 0 .. 500]

TypeOK ≜
    ∧   replicas ∈ Replica
    ∧   pending_counters ∈ [PendingKey → PendingInfo]

initCounter ≜ [count ↦ 0, need_update ↦ FALSE, version ↦ 0]

Init ≜
    ∧ replicas = [id ∈ ReplicaID ↦ nil]
    ∧ pending_counters = [k ∈ PendingKey ↦ initCounter]

addReplicaImpl(id, ds, st) ≜
    LET
        new_repl ≜ [ds ↦ ds, status ↦ "pending", storage ↦ st, agg ↦ FALSE]
        key ≜ ⟨ds, st⟩
        old_counter  ≜ pending_counters[key]
        new_counter  ≜ [old_counter EXCEPT !.need_update = TRUE, !.version = @ + 1]
    IN
        ∧ replicas′ = [replicas EXCEPT ![id] = new_repl]
        ∧ pending_counters′ = [pending_counters EXCEPT ![key] = new_counter]

AddReplica(id, ds, st) ≜
    ∧ replicas[id] = nil
    ∧ addReplicaImpl(id, ds, st)
```

```
$updateCounterAfterWritten(r) \triangleq$
    LET
        $k \triangleq \langle r.ds, r.storage \rangle$
    IN
        $pending\_counters' = [$
            $pending\_counters$ EXCEPT $![k] = [$
                $@$ EXCEPT $!.need\_update = $ TRUE, $!.version = @ + 1$
            $]$
        $]$

$UpdateToWritten(id) \triangleq$
    $\land replicas[id] \neq nil$
    $\land replicas' = [replicas$ EXCEPT $![id].status = $ "written"$]$
    $\land updateCounterAfterWritten(replicas[id])$

$replicaHasKey(id, k) \triangleq$
    $\land replicas[id] \neq nil$
    $\land replicas[id].ds = k[1]$
    $\land replicas[id].storage = k[2]$

$getPendingReplicas(k) \triangleq$
    LET
        $selectCond(id) \triangleq$
            $\land replicaHasKey(id, k)$   TODO missing cond
    IN
        $\{id \in ReplicaID : selectCond(id)\}$

$setAggTrue(update\_ids) \triangleq$
    LET
        $new\_fn(id) \triangleq$
            IF $id \in update\_ids$
                THEN $[replicas[id]$ EXCEPT $!.agg = $ TRUE$]$
                ELSE $replicas[id]$   unchanged
    IN
        $replicas' = [id \in ReplicaID \mapsto new\_fn(id)]$

$doUpdatePendingCounter(k) \triangleq$
    LET
        $pending\_repls \triangleq getPendingReplicas(k)$
        $num \triangleq Cardinality(pending\_repls)$

        $old\_counter \triangleq pending\_counters[k]$
        $new\_counter \triangleq [old\_counter$ EXCEPT $!.count = num, !.need\_update = $ FALSE$]$
    IN
        $\land pending\_counters' = [pending\_counters$ EXCEPT $![k] = new\_counter]$
        $\land setAggTrue(pending\_repls)$

$UpdatePendingCounter(k) \triangleq$
    $\land pending\_counters[k].need\_update = \text{TRUE}$
    $\land doUpdatePendingCounter(k)$

$TerminateCond \triangleq$
    $\land \forall id \in ReplicaID :$
        $\land replicas[id] \neq nil$
        $\land replicas[id].agg = \text{TRUE}$
    $\land \forall key \in PendingKey : pending\_counters[key].need\_update = \text{FALSE}$

$Terminated \triangleq$
    $\land TerminateCond$
    $\land \text{UNCHANGED } vars$

$Next \triangleq$
    $\lor \exists id \in ReplicaID, ds \in Dataset, st \in Storage :$
        $AddReplica(id, ds, st)$
    $\lor \exists id \in ReplicaID :$
        $UpdateToWritten(id)$
    $\lor \exists k \in PendingKey :$
        $UpdatePendingCounter(k)$
    $\lor Terminated$

$allPendingReplicas(k) \triangleq$
    LET
        $checkCond(id) \triangleq$
            $\land replicaHasKey(id, k)$
            $\land replicas[id].status = \text{“pending”}$
        $S \triangleq \{id \in ReplicaID : checkCond(id)\}$
    IN
        $Cardinality(S)$

$numPendingByCounter(k) \triangleq$
    LET
        $checkCond(id) \triangleq$
            $\land replicaHasKey(id, k)$
            $\land replicas[id].agg = \text{FALSE}$
            $\land replicas[id].status = \text{“pending”}$
        $S \triangleq \{id \in ReplicaID : checkCond(id)\}$
    IN
        $Cardinality(S) + pending\_counters[k].count$
```

$Inv \triangleq$

$\quad \land \forall\, k \in PendingKey :$

$\qquad allPendingReplicas(k) = numPendingByCounter(k)$

$Sym \triangleq Permutations(Dataset) \cup Permutations(Storage)$