```
┌──────────────────── MODULE incremental_update ────────────────────┐
EXTENDS Integers, Bags, TLC

CONSTANTS Groups, Users, Documents
VARIABLES userGroups, dUserGroups, docGroups, dDocGroups, mUserDoc

RECURSIVE SumAcc(_, _, _)
SumAcc(f, S, Acc) ≜
   IF S = {} THEN Acc
    ELSE LET x ≜ CHOOSE x ∈ S : TRUE
          IN   SumAcc(f, S \ {x}, Acc + f[x])
Sum(f, S) ≜ SumAcc(f, S, 0)
SumF(F) ≜ Sum(F, DOMAIN F)

Abs(n) ≜ IF n < 0 THEN ( − n) ELSE n
NBag(B) ≜
     LET nonEmpty ≜ {e ∈ DOMAIN B : B[e] ≠ 0}
     IN   [e ∈ nonEmpty ↦ B[e]]

BagPlus(B1, B2) ≜
   [e ∈ (DOMAIN B1) ∪ (DOMAIN B2) ↦
      (IF e ∈ DOMAIN B1 THEN B1[e] ELSE 0)
    + (IF e ∈ DOMAIN B2 THEN B2[e] ELSE 0)]

BagMinus(B1, B2) ≜
   [e ∈ (DOMAIN B1) ∪ (DOMAIN B2) ↦
      (IF e ∈ DOMAIN B1 THEN B1[e] ELSE 0)
    − (IF e ∈ DOMAIN B2 THEN B2[e] ELSE 0)]

BagEq(B1, B2) ≜
   DOMAIN (NBag(BagMinus(B1, B2))) = {}

NoNegativeValues(B) ≜
   ∀ e ∈ DOMAIN B : B[e] ≥ 0

├───────────────────────────────────────────────────────────────────┤

TypeOK ≜
   ∧  (DOMAIN userGroups) ⊆ (Users × Groups)
   ∧  NoNegativeValues(userGroups)
   ∧  (DOMAIN dUserGroups) ⊆ (Users × Groups)

   ∧  (DOMAIN docGroups) ⊆ (Documents × Groups)
   ∧  NoNegativeValues(docGroups)
   ∧  (DOMAIN dDocGroups) ⊆ (Documents × Groups)

   ∧  (DOMAIN mUserDoc) ⊆ (Users × Documents)
   ∧  NoNegativeValues(mUserDoc)
```

A cross product on bags

```
Join(R, S) ≜ [
    ⟨r, s⟩ ∈ (DOMAIN R) × (DOMAIN S) ↦
    CopiesIn(r, R) ∗ CopiesIn(s, S)
]

UserDoc(ugs, gds) ≜ [
    ⟨user, doc⟩ ∈ (Users × Documents) ↦
    LET prod ≜ Join(ugs, gds)
    IN   SumF(NBag([
             ⟨ug, gd⟩   ∈ DOMAIN prod ↦
             IF  ∧ user = ug[1]
                 ∧ ug[2] = gd[1]
                 ∧ gd[2] = doc
              THEN prod[⟨ug, gd⟩] ELSE  0
    ]))
]

AddWithDelta(B, dB, t) ≜
    ∧ B′ = [B EXCEPT ![t] = CopiesIn(t, B) + 1]
    ∧ dB′ = [dB EXCEPT ![t] = CopiesIn(t, dB) + 1]

RemoveWithDelta(B, dB, t) ≜
    ∧ B[t] ≠ 0
    ∧ B′ = [B EXCEPT ![t] = @ − 1]
    ∧ dB′ = [dB EXCEPT ![t] = @ − 1]

AddUserToGroup ≜ ∃ t ∈ (Users × Groups) :
    ∧ AddWithDelta(userGroups, dUserGroups, t)
    ∧ UNCHANGED ⟨docGroups, dDocGroups, mUserDoc⟩

RemoveUserFromGroup ≜ ∃ t ∈ DOMAIN userGroups :
    ∧ RemoveWithDelta(userGroups, dUserGroups, t)
    ∧ UNCHANGED ⟨docGroups, dDocGroups, mUserDoc⟩

PublishDocumentForGroup ≜ ∃ doc ∈ Documents, group ∈ Groups :
    ∧ AddWithDelta(docGroups, dDocGroups, ⟨doc, group⟩)
    ∧ UNCHANGED ⟨userGroups, dUserGroups, mUserDoc⟩

HideDocumentFromGroup ≜ ∃ t ∈ DOMAIN docGroups :
    ∧ RemoveWithDelta(docGroups, dDocGroups, t)
    ∧ UNCHANGED ⟨userGroups, dUserGroups, mUserDoc⟩
```

delta *mUserDoc* na podstawie delt *userGroups i docGroups*

```
UserDocDelta(dUG, dDG) ≜
    BagMinus(
        BagPlus(UserDoc(dUG, docGroups),
                UserDoc(userGroups, dDG)),
        UserDoc(dUG, dDG))

 UserDoc(userGroups, docGroups) − mUserDoc
dmUserDoc ≜
    UserDocDelta(dUserGroups, dDocGroups)

ApplyAllDeltas ≜
    ∧ mUserDoc′ = BagPlus(mUserDoc, dmUserDoc)
    ∧ dUserGroups′ = [t ∈ (Users × Groups) ↦ 0]
    ∧ dDocGroups′ = [t ∈ (Documents × Groups) ↦ 0]
    ∧ UNCHANGED ⟨userGroups, docGroups⟩

emptyUserGroups ≜ [t ∈ (Users × Groups) ↦ 0]
emptyDocGroups ≜ [t ∈ (Documents × Groups) ↦ 0]

ApplySomeUserDeltas ≜ ∃ ts ∈ SUBSET (DOMAIN dUserGroups) :
   LET dUG ≜ [t ∈ ts ↦ dUserGroups[t]]IN
    ∧ mUserDoc′ = BagPlus(mUserDoc, UserDocDelta(dUG, emptyDocGroups))
    ∧ dUserGroups′ = BagMinus(dUserGroups, dUG)
    ∧ UNCHANGED ⟨userGroups, docGroups, dDocGroups⟩

ApplySomeDocumentDeltas ≜ ∃ ts ∈ SUBSET (DOMAIN dDocGroups) :
   LET dDG ≜ [t ∈ ts ↦ dDocGroups[t]]IN
    ∧ mUserDoc′ = BagPlus(mUserDoc, UserDocDelta(emptyUserGroups, dDG))
    ∧ dDocGroups′ = BagMinus(dDocGroups, dDG)
    ∧ UNCHANGED ⟨userGroups, docGroups, dUserGroups⟩

deltasEmpty ≜
    ∧ BagEq(dUserGroups, EmptyBag)
    ∧ BagEq(dDocGroups, EmptyBag)

Init ≜
    ∧ userGroups = emptyUserGroups
    ∧ dUserGroups = emptyUserGroups
    ∧ docGroups = emptyDocGroups
    ∧ dDocGroups = emptyDocGroups
    ∧ deltasEmpty
    ∧ mUserDoc = UserDoc(userGroups, docGroups)
    ∧ BagEq(mUserDoc, EmptyBag)

Next ≜
    ∨ AddUserToGroup
    ∨ RemoveUserFromGroup
    ∨ PublishDocumentForGroup
    ∨ HideDocumentFromGroup
    ∨ ApplySomeUserDeltas
    $\lor ApplySomeDocumentDeltas$
    $\lor ApplyAllDeltas$

$Spec \triangleq$
    $\land Init$
    $\land \Box[Next]_{\langle userGroups,\ dUserGroups,\ docGroups,\ dDocGroups,\ mUserDoc \rangle}$
    $\land \Box[(\neg deltasEmpty) \leadsto \text{FALSE}]$

$Consistent \triangleq$
    $\land deltasEmpty \Rightarrow BagEq(mUserDoc,\ UserDoc(userGroups,\ docGroups))$
    $\land BagEq(BagPlus(mUserDoc,\ dmUserDoc),\ UserDoc(userGroups,\ docGroups))$

$MaxDelta(n) \triangleq$
    $\land n \geq SumF([i \in \text{DOMAIN } dUserGroups \mapsto Abs(dUserGroups[i])])$
    $\land n \geq SumF([i \in \text{DOMAIN } dDocGroups \mapsto Abs(dDocGroups[i])])$

$MaxDups(n) \triangleq$
    $\land \quad \forall t \in \text{DOMAIN } userGroups : userGroups[t] \leq n$
    $\land \quad \forall t \in \text{DOMAIN } docGroups : docGroups[t] \leq n$

 $\rightarrow R,\ dR,\ S,\ dS$
 $\leftarrow$ Hop, $dHop$, $TriHop$, $dTriHop$
always: $dR$ is empty $\land$ $dS$ is empty $\Rightarrow$ Hop \eq $vHop$
always: $dR$ is empty $\land$ $dS$ is empty $\land$ $dHop$ is empty $\Rightarrow$ $TriHop$ \eq $vTriHop$
eventually: $dR$ is empty $\land$ $dS$ is empty $\land$ $dHop$ is empty
```

\ * Modification History
\ * Last modified Sun *Dec* 16 12:48:37 *CET* 2018 by *tomek*
\ * Created *Wed Dec* 12 22:55:34 *CET* 2018 by *tomek*