Eidgenössische Technische Hochschule Zürich
Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Swiss Federal Institute of Technology Zurich

Institut für Informationssysteme, Dr. Markus Tresch — Uwe Röhm

Verteilte Objektverwaltung: Client, Server & Middleware
Lösungsvorschlag Übung 5

Aufgabe 1: Selbstwartbarkeit einer Data Warehouse Sicht

Datenbank	Operation	selbstwartbar
BusinessDB	Create(...)	—
	Delete(*obj*)	ja
	Update(...)(1)	—
PersonalDB	Create(...)	—
	Delete(*obj*)	ja
	Update(...)(2)	—
InventoryDB	Create(...)	—
	Delete(*obj*)	—
	Update(...)(3)	—

(1) Falls sich der von `Classify()` berechnete Wert dabei verändert, wenn die Attribute `From` oder `To` geändert werden. Änderungen an anderen Attributen sind selbstwartbar, können also ignoriert werden.
(2) Nur Änderungen auf `Licenses` sind wichtig.
(3) Nur Änderungen in `Classes` oder von `Model` sind relevant.

Aufgabe 2: Sichtenwartung

Änderungsoperationen in der BusinessDB

```
on create obj(Name, From, To) in Flights@BusinessDB do
   create (Name, From, To, Pilots := ( select p
                                       from  Pilots@PersonalDB p,
                                             Planes@InventoryDB a
                                       where Classify(obj) in Usages(a)
                                       and   Model(a) in Licenses(p) ))
   in Connections

on delete (obj) in Flights@BusinessDB do
   delete obj from Connections

on update (obj, From, old, new) in Flights@BusinessDB do
   set Pilots(obj) := ( select p
                        from  Pilots@PersonalDB p,
                              Planes@InventoryDB a
                        where Classify(obj) in Usages(a)
                        and   Model(a) in Licenses(p) )
   from Connections
```

update (obj, To, old, new) äquivalent ...

Änderungsoperationen in der PersonalDB

```
on create (obj, PNr, Name, Adr, Licenses) in Pilots@PersonalDB do
   add      (obj, Pilots(c))
   from     Connections c,
            Planes@InventoryDB a
   where    Model(a) in Licenses
   and      Classify(c) in Usages(a)

on delete (obj) in Pilots@PersonalDB do
   remove   (obj, Pilots(c))
   from     Connections c
   where    obj in Pilots(c)

on update (obj, Licenses, add_op, new) in Pilots@PersonalDB do
   add      (obj, Pilots(c))
   from     Connections c,
            Planes@InventoryDB a
   where    not(obj in Pilots(c))
   and      Classify(c) in Usages(a)
   and      Model(a) = new

on update (obj, Licenses, remove_op, old) in Pilots@PersonalDB do
   set      Pilots(c) := ( select p
                           from  Pilots@PersonalDB p,
                                 Planes@InventoryDB a
                           where Classify(c) in Usages(a)
                           and   Model(a) in Licenses(p) )
   from     Connections c
   where    obj in Pilots(c)
```

Änderungsoperationen in der InventoryDB

```
on create (obj, InvNr, Model, Manufacturer, Seats, Usages) in Planes@InventoryDB do
   add      (p, Pilots(c))
   from     Connections c,
            Pilots@PersonalDB p
   where    not(p in Pilots(c))
   and      Classify(c) in Usages
   and      Model in Licenses(p)

on delete (obj) in Planes@InventoryDB do
   set      Pilots(c) := ( select p
                           from  Pilots@PersonalDB p,
                                 Planes@InventoryDB a
                           where Classify(c) in Usages(a)
                           and   Model(a) in Licenses(p) )
   from     Connections c
   where    Classify(c) in Usages(obj)

on update (obj, Model, old, new) in Planes@InventoryDB do
   set      Pilots(c) := ( select p
                           from  Pilots@PersonalDB p,
```

```
                                Planes@InventoryDB a
                          where Classify(c) in Usages(a)
                          and   new in Licenses(p) )
    from      Connections c
    where     Classify(c) in Usages(obj)

on update (obj, Usages, add_op, new) in Planes@InventoryDB do
    add       (p, Pilots(c))
    from      Connections c,
              Pilots@PersonalDB p
    where     not(p in Pilots(c))
    and       Classify(c) in Usages
    and       Model(obj) in Licenses(p)

on update (obj, Usages, remove_op, old) in Planes@InventoryDB do
    set       Pilots(c) := ( select p
                          from  Pilots@PersonalDB p,
                                Planes@InventoryDB a
                          where Classify(c) in Usages(a)
                          and   new in Licenses(p) )
    from      Connections c
    where     Classify(c) in Usages(obj)
```

Aufgabe 3: Zähler-Algorithmus

Wir führen einen Zähler `count` an den Pilot-Objekten im Attribut `Pilots` der Sicht `Connections` ein, das angibt, wie häufig ein Pilot-Objekt sich für diese Menge qualifiziert hat. Diesen Zähler erniedrigen wir, sobald einem Pilot der PersonalDB eine Lizenz entzogen wird, die für diese Verbindung relevant ist. Wenn der Zähler auf 0 heruntergezählt wurde, so bedeutet dies, daß dieser Pilot die entsprechende Verbindung mit keiner seiner verbleibenden Lizenzen fliegen darf. Er kann aus der `Pilots` Menge entfernt werden:

```
on update (obj, Licenses, remove_op, old) in Pilots@PersonalDB do
    decrement (count_obj, Pilots(c))
    from      Connections c,
              Planes@InventoryDB a
    where     obj in Pilots(c)
    and       Classify(c) in Usages(a)
    and       Model(a) = old
    on        count_obj = 0 remove (obj, Pilots(c))
```

Das gleiche gilt für das Löschen eines Flugzeug-Objektes und das Entfernen eines Eintrags aus den Verwendungsmöglichkeiten `Usages` eines Flugzeugs in der InventoryDB:

```
on delete (obj) in Planes@InventoryDB do
    decrement (count_p, Pilots(c))
    from      Connections c,
              Pilots@PersonalDB p
    where     p in Pilots(c)
    and       Classify(c) in Usages(obj)
    and       Model(obj) in Licenses(p)
    on        count_p = 0 remove (p, Pilots(c))
```

```
on update (obj, Usages, remove_op, old) in Planes@InventoryDB do
   decrement (count_p, Pilots(c))
   from      Connections c,
             Pilots@PersonalDB p
   where     p in Pilots(c)
   and       Classify(c) = old
   and       Model(obj) in Licenses(p)
   on        count_p = 0 remove (p, Pilots(c))
```

Leider wird die Sicht `Connections` somit *nicht* selbstwartbar bezüglich dieser beiden Änderungsoperationen. Um sicherzustellen, daß die jeweilige Änderung wirklich relevant für eine bestimmte Verbindung ist, muß nach wie vor im anderen Datenbestand nachgeschaut werden. Zudem müssen in den anderen Operationen die Zähler entsprechend mit gewartet werden. Die Selbstwartbarkeit kann erst dann erweitert werden, wenn Piloten- und/oder Flugzeug-Objekte ebenfalls im Data Warehouse materialisiert werden.