Περιορισμοί ακεραιότητας

Στα παρακάτω επιλέξαμε να μην αλλάζουμε ποτέ το πρωτεύον κλειδί των πλειάδων, μιας και θεωρήσαμε ότι δεν υπάρχει λόγος να γίνεται αυτό, και απλά θα επέφερε επιπλοκές.

**Member**

**-**Όταν δημιουργείται ένα νέο μέλος, πρέπει να έχει μηδενικό **balance, debt, totalMoneySpent**. Επίσης, πρέπει το τηλέφωνό του, **tel**, να είναι ακριβώς δέκα ψηφία. Στη βάση αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `member\_domain\_constraints` BEFORE INSERT ON `member`

FOR EACH ROW BEGIN

IF (NEW.tel REGEXP '[0-9]{10}') = 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid phone number!';

END IF;

IF NEW.debt != 0 THEN

SET NEW.debt = 0;

END IF;

IF NEW.totalMoneySpent != 0 THEN

SET NEW.totalMoneySpent = 0;

END IF;

IF NEW.balance != 0 THEN

SET NEW.balance = 0;

END IF;

END

, το οποίο διασφαλίζει αν δοθούν μη μηδενικά **balance, debt, totalMoneySpent,** στη βάση θα εισαχθούν μηδενικά τέτοια γνωρίσματα, ενώ αν δοθεί άκυρο τηλέφωνο ειδοποιεί κατάλληλα με μήνυμα λάθους.

- Όταν γίνονται update τα στοιχεία ενός μέλους, δεν πρέπει ποτέ να αλλάξει το **mid,** όντας πρωτεύον κλειδί, ούτε το **username**, όντας επίσης αναγνωριστικό, ενώ δεν πρέπει να δοθεί άκυρο τηλέφωνο, ή αρνητικά **debt, balance, totalMoneySpent.** Στη βάση αυτό ελέγχεται με το εξής trigger:

REATE TRIGGER `member\_update\_constraints` BEFORE UPDATE ON `member`

FOR EACH ROW BEGIN

IF (NEW.tel REGEXP '[0-9]{10}') = 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid phone number!';

END IF;

IF NEW.debt < 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid debt!';

END IF;

IF NEW.totalMoneySpent < 0 THEN

signal sqlstate '45000' set message\_text = 'Invalid total money spent for member!';

END IF;

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Member id cannot be changed';

END IF;

IF NEW.username != OLD.username THEN

signal sqlstate '45000' set message\_text = 'Username cannot be changed';

END IF;

IF NEW.balance < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative balance!';

END IF;

END

-Ένα μέλος μπορεί να σβηστεί μόνο εφόσον δεν έχει χρέος και έχει παραλάβει όλες τις παραγγελίες του. Αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `member\_delete\_constraints` BEFORE DELETE ON `member`

FOR EACH ROW BEGIN

IF OLD.debt != 0 THEN

signal sqlstate '45000' set message\_text = 'Cannot delete a member with debt';

END IF;

IF EXISTS(SELECT state FROM order\_t o WHERE o.mid = OLD.mid AND o.state != 'DELIVERED') THEN

signal sqlstate '45000' set message\_text = 'Cannot delete a member with undelivered orders';

END IF;

END

-Όταν εισάγεται ένα μέλος στον **Member**, πρέπει να εισαχθεί το **mid** σε ΑΚΡΙΒΩΣ έναν από τους πίνακες **Client, Merchant** (βλέπε σχέση IsA). Ο έλεγχος αυτός επιλέξαμε να γίνεται server-side, δηλαδή ο χρήστης της DB να εισάγει το μέλος στον πίνακα που αυτός αποφασίζει.

**Client**

**­**-Άπαξ και εισαχθεί ένα μέλος στον **Client**, δεν θα πρέπει ποτέ να αλλάζει το **mid** του, ως πρωτεύον κλειδί. Αυτό ελέγχεται με το εξής trigger:

CREATE TRIGGER `client\_update\_constraints` BEFORE UPDATE ON `client`

FOR EACH ROW BEGIN

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Member id cannot be changed';

END IF;

END

-Το **mid,** εκτός απο κλειδί της **Client**, είναι και ξένο κλειδί που αναφέρεται στην **Member,** οπότε κάθε φορά που διαγράφεται ένα στοιχείο της **Member** πρέπει να διαγράφεται και το αντίστοιχο στοιχείο της **Client**, αν υπάρχει. Αυτό είναι ένας περιορισμός αναφοράς, και ελέγχεται μέσω χρήσης foreign key constraint στην phpmyadmin, που ορίζει ότι κατά τη διαγραφή στοιχείου από την **Member,** θα ακολουθείται cascaded deletion policy.

ALTER TABLE `client`

ADD CONSTRAINT `FK\_Client\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE CASCADE ON UPDATE CASCADE;

**Merchant**

Τα πράγματα εδώ είναι ακριβώς αντίστοιχα με τον **Client.**

ALTER TABLE `merchant`

ADD CONSTRAINT `FK\_Merchant\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE CASCADE ON UPDATE CASCADE;

**Order**

-Δεν επιτρέπεται ποτέ μια **Order** να έχει αρνητικό κόστος, **state** που δεν ανήκει στο {FULLY\_PAID, DELIVERED, AWAITING\_PAYMENT} ή **discount** που δεν ανήκει στο {0, 5, 10}. Επίσης δεν επιτρέπεται να αλλάζει το **oid** ή το **mid** μετά τη δημιουργία της παραγγελίας. Αυτά ελέγχονται με τα εξής δύο triggers:

CREATE TRIGGER `order\_t\_update\_constraints` BEFORE UPDATE ON `order\_t`

FOR EACH ROW BEGIN

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.mid != OLD.mid THEN

signal sqlstate '45000' set message\_text = 'Cannot change member id referred to by order after creation of order';

END IF;

IF NEW.state != 'DELIVERED' AND NEW.state != 'AWAITING\_PAYMENT' AND NEW.state != 'FULLY\_PAID' THEN

signal sqlstate '45000' set message\_text = 'Invalid order state';

END IF;

IF NEW.cost < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative order cost';

END IF;

IF NEW.discount != 0 AND NEW.discount != 5 AND NEW.discount != 10 THEN

signal sqlstate '45000' set message\_text = 'Invalid order discount';

END IF;

END

- Η **Order** έχει ξένο κλειδί το **mid**, άρα έχουμε περιορισμό αναφοράς (foreign key constraint στην phpmyadmin), και επιλέγουμε την εξής πολιτική: όταν ένα μέλος διαγράφεται, οι παραγγελίες του παραμένουν στο σύστημα ώστε να είναι διαθέσιμες οι πληροφορίες σχετικά με το πόσα κρασιά κάθε είδους έχουν πουληθεί αυτόν τον μήνα. Άρα, θέτουμε το **mid** της **Order** ως null.

ALTER TABLE `order\_t`

ADD CONSTRAINT `FK\_Order\_t\_Member` FOREIGN KEY (`mid`) REFERENCES `member` (`mid`) ON DELETE SET NULL;

**Wine**

-Ανά πάσα στιγμή απαγορεύεται ένα κρασί να έχει αρνητική τιμή λιανικής ή χονδρικής πώλησης, ενώ το **wid** δεν επιτρέπεται να αλλάζει. Υπάρχουν τα εξής δύο triggers στη βάση:

CREATE TRIGGER `wine\_domain\_constraints` BEFORE INSERT ON `wine`

FOR EACH ROW BEGIN

IF NEW.retailPrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative retail price!';

END IF;

IF NEW.wholesalePrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative wholesale price!';

END IF;

END

REATE TRIGGER `wine\_update\_constraints` BEFORE UPDATE ON `wine`

FOR EACH ROW BEGIN

IF NEW.retailPrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative retail price!';

END IF;

IF NEW.wholesalePrice <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative wholesale price!';

END IF;

IF NEW.wid != OLD.wid THEN

signal sqlstate '45000' set message\_text = 'Wine id cannot be changed';

END IF;

END

**MostPopularWine**

-Το **bottlesThisMonth** δεν επιτρέπεται να είναι αρνητικό, ούτε και να αλλάζει μετά τη δημιουργία της πλειάδας. Υπάρχουν στη βάση τα εξής δύο triggers:

CREATE TRIGGER `mostpopularwines\_domain\_constraints` BEFORE INSERT ON `mostpopularwines`

FOR EACH ROW BEGIN

IF NEW.bottlesThisMonth <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

END

CREATE TRIGGER `mostpopularwines\_update\_constraints` BEFORE UPDATE ON `mostpopularwines`

FOR EACH ROW BEGIN

IF NEW.bottlesThisMonth != OLD.bottlesThisMonth THEN

signal sqlstate '45000' set message\_text = 'Can''t change number of bottles in mostpopularwines after creation';

END IF;

END

-Επίσης, το **wid** είναι και ξένο κλειδί στην **mostPopularWine**, άρα έχουμε έναν περιορισμό αναφοράς στον πίνακα **Wine.** Επιλέξαμε να μην σβήνεται ποτέ ένα κρασί από τον πίνακα **Wine** άπαξ και εισαχθεί στη βάση**,** άρα ακολουθήσαμε restriction policy στη διαγραφή.

ALTER TABLE `mostpopularwines`

ADD CONSTRAINT `FK\_MostPopularWines` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`);

**Variety**

Δεν έχει περιορισμούς ακεραιότητας.

**Transaction**

-Το **type** δεν μπορεί να πάρει τιμή διαφορετική των **PAYMENT, REFUND**, ενώ το **amount** δεν μπορεί να γίνει ποτέ αρνητικό. Επίσης, δεν πρέπει ποτέ μια νέα δοσοληψία μιας παραγγελίας να έχει ημερομηνία παλαιότερη από την ίδια την παραγγελία. Τα εξής δύο triggers υπάρχουν στη βάση:

CREATE TRIGGER `transactions\_domain\_constraints` BEFORE INSERT ON `transactions`

FOR EACH ROW BEGIN

IF NEW.type <> 'PAYMENT' && NEW.type <> 'REFUND' THEN

signal sqlstate '45000' set message\_text = 'Invalid state!';

END IF;

IF NEW.amount < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount!';

END IF;

IF NEW.date < (SELECT date from order\_t WHERE order\_t.oid = NEW.oid) THEN

signal sqlstate '45000' set message\_text = 'Transaction date older than corresponding order!';

END IF;

END

CREATE TRIGGER `transactions\_update\_constraints` BEFORE UPDATE ON `transactions`

FOR EACH ROW BEGIN

IF NEW.type <> 'PAYMENT' && NEW.type <> 'REFUND' THEN

signal sqlstate '45000' set message\_text = 'Invalid state!';

END IF;

IF NEW.amount < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount!';

END IF;

IF NEW.date < (SELECT date from order\_t WHERE order\_t.oid = NEW.oid) THEN

signal sqlstate '45000' set message\_text = 'Transaction date older than corresponding order!';

END IF;

END

-Το **oid** είναι ξένο κλειδί, άρα έχουμε έναν περιορισμό αναφοράς. Επιλέγουμε cascaded deletion policy, αφού όταν διαγράφεται μια παραγγελία θέλουμε να διαγράφονται και όλα τα σχετικά transactions.

ALTER TABLE `transactions`

ADD CONSTRAINT `FK\_Payment\_Order` FOREIGN KEY (`oid`) REFERENCES `order\_t` (`oid`) ON DELETE CASCADE;

**OrderConsistsOfWine**

**-**Το γνώρισμα **amount** δεν επιτρέπεται να πάρει αρνητική τιμή. Επίσης, δεν επιτρέπεται να αλλάζει η τιμή των **oid, wid** μετά τη δημιουργία μιας πλειάδας. Υπάρχουν τα εξής triggers:

CREATE TRIGGER `orderconsistsofwine\_domain\_constraints` BEFORE INSERT ON `orderconsistsofwine`

FOR EACH ROW BEGIN

IF NEW.amount <= 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

END

CREATE TRIGGER `orderconsistsofwine\_update\_constraints` BEFORE UPDATE ON `orderconsistsofwine`

FOR EACH ROW BEGIN

IF NEW.amount < 0 THEN

signal sqlstate '45000' set message\_text = 'Negative amount of bottles!';

END IF;

IF NEW.amount > OLD.amount THEN

signal sqlstate '45000' set message\_text = 'Cannot increase order''s bottles after submission';

END IF;

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.oid != OLD.oid THEN

signal sqlstate '45000' set message\_text = 'Cannot change order id after creation of order';

END IF;

IF NEW.wid != OLD.wid THEN

signal sqlstate '45000' set message\_text = 'Cannot change id of wine referred by order after creation of order';

END IF;

END

-Υπάρχουν επίσης δύο ξένα κλειδιά, **oid & wid**, άρα δύο περιορισμοί αναφοράς. Όταν διαγράφεται μια παραγγελία διαγράφονται και όλα τα σχετικά **OrderConsistsOfWine** άρα έχουμε cascaded deletion policy, ενώ εφόσον έχουμε την παραδοχή ότι κρασιά δεν θα διαγράφονται, έχουμε restrict deletion policy για το **wid.**

ALTER TABLE `orderconsistsofwine`

ADD CONSTRAINT `FK\_Wine` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`),

ADD CONSTRAINT `FK\_order` FOREIGN KEY (`oid`) REFERENCES `order\_t` (`oid`) ON DELETE CASCADE;

**WineMadeOfVariety**

-Έχουμε δύο ξένα κλειδιά, **wid & oid**, άρα δύο περιορισμούς αναφοράς. Επειδή ποικιλίες και κρασιά θεωρούμε ότι δεν θα διαγράφονται, βάζουμε restrict deletion policy.

ALTER TABLE `winemadeofvariety`

ADD CONSTRAINT `FK\_Variety` FOREIGN KEY (`vid`) REFERENCES `variety` (`vid`),

ADD CONSTRAINT `FK\_WineMadeOfVariety\_Wine` FOREIGN KEY (`wid`) REFERENCES `wine` (`wid`);

Συναρτησιακές εξαρτήσεις

**Member**

**mid -> username, fname, lname, tel, address, city, country, debt, cardno, balance, password, totalMoneySpent**

**username->mid, fname, lname, tel, address, city, country, debt, cardno, balance, password, totalMoneySpent**

(αφού **mid, username** κλειδιά)

**MostPopularWines**

**Wid->bottlesThisMonth**

(αφού **wid** κλειδί)

**OrderConsistsOfWine**

**oid wid -> amount**

(αφού {**oid, wid**} κλειδί)

**Order\_t**

**Oid -> mid, date, cost, state**

(αφού **oid** κλειδί)

**Transaction**

**Oid date -> type, amount**

, αφού στην **Transaction** ο συνδυασμός των **oid, date** προέκυψε ως μερικό αναγνωριστικό (και επομένως δρα τρόπον τινά ως κλειδί)

**Variety**

**Vid->name**

**Name->vid**

, αφού και τα 2 γνωρίσματα είναι κλειδιά

**Wine**

**Wid -> retailPrice, wholesalePrice, winery, name, color, date, photo**

**retailPrice, wholesalePrice, winery, name, color, date -> wid, photo**

(αφού **wid** κλειδί και αφού δεν μπορούν να υπάρχουν στη βάση δύο κρασιά που διαφέρουν μόνο κατά το **wid**)

**WineMadeOfVariety**

Δεν υπάρχουν μη τετριμμένες συναρτησιακές εξαρτήσεις (αφού η σχέση έχει ως γνωρίσματα μόνο τα **wid, vid** των οποίων το σύνολο αποτελεί και το πρωτεύον κλειδί της σχέσης)

Καθορισμός κλειδιών σχέσεων από συναρτησιακές εξαρτήσεις

Λόγω της απλότητας των συναρτησιακών εξαρτήσεων, τα μοναδικά υποψήφια κλειδιά είναι τα όσα έχουν αναφερθεί στην ενότητα «Πρωτεύοντα κλειδιά», ενώ στο **Wine** υποψήφιο κλειδί είναι και το σύνολο { **retailPrice, wholesalePrice, winery, name, color, date**}. Επειδή λοιπόν δεν υπάρχουν άλλες (ουσιώδεις) συναρτησιακές εξαρτήσεις, ο αλγόριθμος εύρεσης υποψήφιων κλειδιών δεν δίνει άλλα κλειδιά εκτός των προαναφερθέντων.

Μετατροπή βάσης σε 3η κανονική μορφή

Μια σχέση R με σύνολο συναρτησιακών εξαρτήσεων F είναι σε 3η κανονική μορφή αν για κάθε μη τετριμμένη συναρτησιακή εξάρτηση Χ->Α που ανήκει στο F+ ισχύει ότι:

Χ είναι υπερκλειδί της R, ή

Α είναι ένα πρωτεύον γνώρισμα της R

Για κάθε μία από τις σχέσεις στη βάση μας, οι συναρτησιακές εξαρτήσεις που υπάρχουν είναι αρκούντως απλές και λίγες ώστε το κλείσιμο του εκάστοτε συνόλου τους να είναι το ίδιο το σύνολο. Επειδή λοιπόν αυτό το σύνολο είναι μόνο εξαρτήσεις της μορφής Χ->Α όπου Χ κλειδί της R, το Χ είναι και υπερκλειδί, άρα ικανοποιείται η πρώτη από τις προαναφερθείσες προϋποθέσεις ώστε η R να είναι σε 3η κανονική μορφή.

Άρα, όλες οι σχέσεις της βάσης είναι σε 3η κανονική μορφή, οπότε και το σχήμα της βάσης δεδομένων μας είναι ήδη σε 3η κανονική μορφή και δεν χρειάζεται αλλαγές.