ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ & ΜΗΧΑΝΙΚΩΝ Η/Υ
ΤΟΜΕΑΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

Project στα πλαίσια του μαθήματος
Βάσεις Δεδομένων

«*Ανάπτυξη εφαρμογής για τα Μικροδάνεια*»

Ομάδα 13:
Καραμπάση Αικατερίνη, 03112517
Μανδηλαράς Νικηφόρος, 03112012
Πασχάλης Ηλίας, 03112122

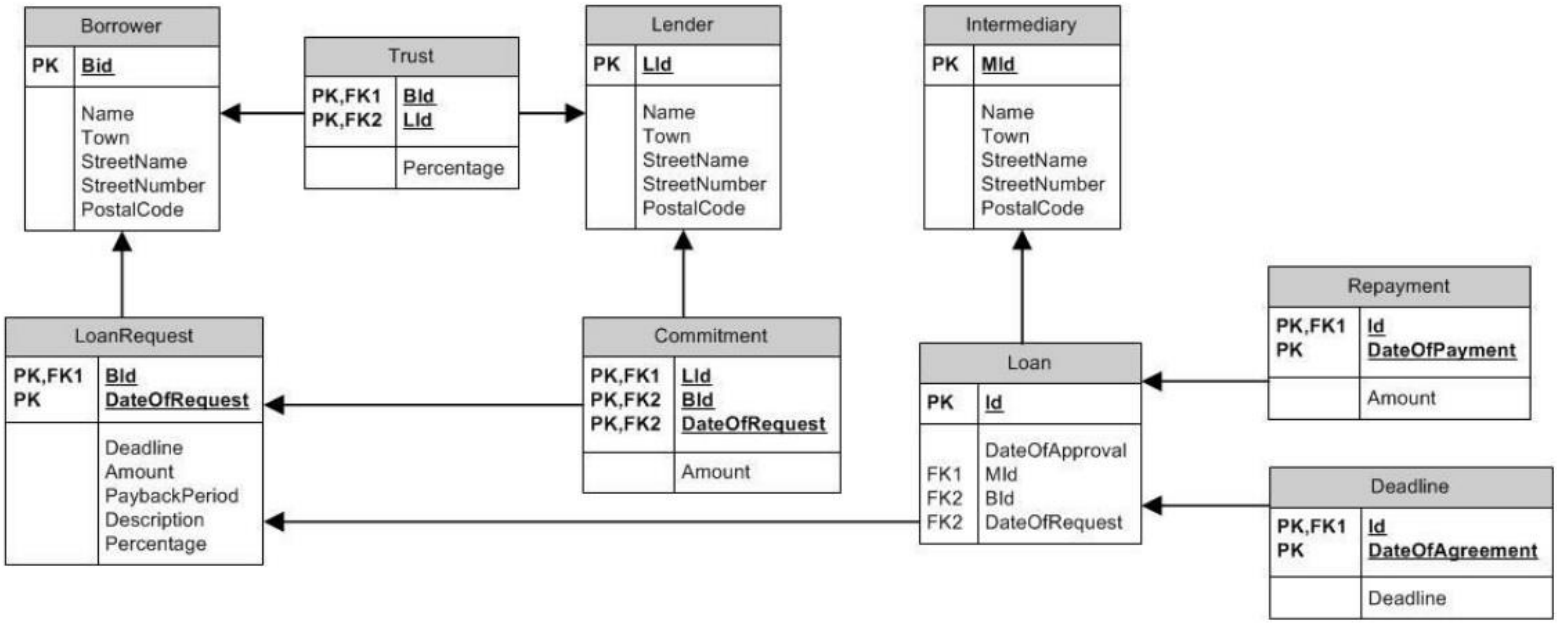
ΑΘΗΝΑ, 2016

Εισαγωγή

Στην εργασία αυτή μας ζητήθηκε να υλοποιήσουμε τη βάση δεδομένων που θα αξιοποιήσει μία τράπεζα για τα μικροδάνεια τα οποία χειρίζεται. Οι σχέσεις που ικανοποιούνται μεταξύ των ενδιαφερόμενων δόθηκαν ως δεδομένα στην πρώτη εργασία. Αυτές είναι:

- Για κάθε άνθρωπο που εμπλέκεται σε έκαστο μικροδάνειο κρατάμε ορισμένα στοιχεία του.
- Κάθε δάνειο δίνεται έπειτα από αίτησή του, αναγράφοντας πληροφορίες σχετικά με το λόγο αίτησής του, την περίοδο αποπληρωμής του αλλά και το επιτόκιό του.
- Μετά την αίτηση, κάποιος Δανειστής μπορεί να δεσμευτεί ότι θα δώσει μέρος του ποσού που ζητείται.
- Μόλις συμπληρωθούν οι δεσμεύσεις από διάφορους Δανειστές στο ποσό του δανείου, τότε αυτό δίνεται. Σε αντίθετη περίπτωση, το αίτημα ακυρώνεται. Ο δανειζόμενος μπορεί να κάνει πολλές αιτήσεις για δάνεια, όχι όμως την ίδια μέρα.
- Για την ασφάλεια του δικτύου οι πληρωμές γίνονται μέσω Μεσάζοντα.
- Ο Δανειζόμενος επιλέγει πότε θα πληρώσει κάποια δόση του δανείου. Ωστόσο, ο ίδιος Δανειζόμενος δεν μπορεί να πληρώσει περισσότερες της μίας δόσης ανά δάνειο στην ίδια μέρα. Για κάθε Δόση ο Δανειστής θα πληρωθεί αναλογικά με το ποσοστό συμμετοχής του στο Δάνειο.
- Εάν το δάνειο δεν αποπληρωθεί στη συμφωνηθείσα ημερομηνία τότε μία νέα ημερομηνία συμφωνείται.
- Για κάθε Δανειζόμενο ο Δανειστής κρατάει ένα δείκτη εμπιστοσύνης, ο οποίος αντανακλά το βαθμό φερεγγυότητας του Δανειζόμενου.

Με βάση αυτές τις προδιαγραφές μας δόθηκε μία λύση στην εργασία η οποία ικανοποιεί αυτό το σύστημα βάσεων. Μία σχηματική απεικόνιση των σχέσεων που συνδέουν τους ρόλους Δανειστής, Δανειζόμενος, Δάνειο, Αίτηση, Συμφωνία, Δείκτης Εμπιστοσύνης μας δόθηκε ως ενδεικτική λύση στο παραπάνω πρόβλημα και φαίνεται παρακάτω:



Για την υλοποίηση της βάσης χρησιμοποιήσαμε το πρόγραμμα WAMP. Μέσω αυτού του εργαλείου και αξιοποιώντας τις γνώσεις μας περί της γλώσσας SQL δημιουργήσαμε τη βάση μας, δηλαδή τους πίνακες με τις σχέσεις που τις περιβάλλουν. Η επιλογή του συγκεκριμένου προγράμματος έγινε αποκλειστικά με το σκεπτικό τη διευκόλυνση ως προς το σχεδιασμό της βάσης μας. Πέραν της βάσης, χρειάστηκε να δημιουργήσουμε και ένα User Interface με σκοπό τη διευκόλυνση του χρήστη ως προς την αξιοποίηση της εφαρμογής μας. Για τη δημιουργία της εφαρμογής μας χρησιμοποιήσαμε τις γλώσσες HTML και PHP, για το interface και τη σύνδεση του site με την εφαρμογή μας, αντίστοιχα. Έτσι, δημιουργήσαμε τα μενού, μέσω των οποίων ο χρήστης μπορεί να χειριστεί τη βάση χωρίς να είναι γνώστης καμία προγραμματιστικής γλώσσας. Με αυτό το σκοπό, η αρχική σελίδα δίνει τις επιλογές Insert, Update, Delete, DB Info, Views και Queries. Επιπλέον, έχει δοθεί η δυνατότητα μέσω διαφόρων dropdown lists να επιλέγουμε ακριβώς κάθε φορά ποια ενέργεια επιθυμούμε να εκτελεστεί, κάνοντας έτσι πιο εύχρηστη την εφαρμογή στο χρήστη. Επίσης, προνοούμε για τις περιπτώσεις που θα γίνει κάποιο λάθος, θεμιτό ή αθέμιτο, αλλά το οποίο μπορεί να δώσει λανθασμένα αποτελέσματα στη βάση μας. Επομένως, προνοούμε ώστε να μην έχουμε αρνητικές τιμές σε ημερομηνίες, Τ.Κ., αλλά και να μην αφήνονται κενά ορισμένα πεδία τα οποία κρίνονται απαραίτητα για τη σωστή συμπλήρωση κάθε πλειάδας κι επομένως των πινάκων της βάσης μας.

Για να δώσουμε μορφή στην ιστοσελίδα μας που να είναι φιλική προς το χρήστη αλλά και να είναι πιο εύχρηστη αξιοποιήσαμε τη μορφοποίηση που εύκολα μας προσφέρει το αρχείο “stylesheet.css”. Το αρχείο αυτό χρησιμοποιεί απλή σύνταξη προκειμένου να δώσει μορφή στο κείμενό μας, όπως είναι η επιλογή της γραμματοσειράς, το μέγεθός της, η μορφοποίηση των μενού επιλογής και συμπλήρωσης, όταν απαιτείται σε κάποια φόρμα, αλλά και ο τρόπος που θα επιλέξουμε να εμφανίζονται τα αποτελέσματά μας. Μέσω των κλάσεων χειριστήκαμε είδη που θέλαμε να τους δώσουμε συγκεκριμένη μορφή αλλά που θέλαμε να εφαρμοστεί σε αυτά μόνο τα αντικείμενα. Στους συνδέσμους οι οποίοι μας μεταφέρουν σε άλλη σελίδα φτιάξαμε με τέτοιο τρόπο την εφαρμογή που να αλλάζουν χρώμα τα γράμματα ανάλογα με το αν βρίσκεται το ποντίκι πάνω από κάποια επιλογή-μεταφορά ώστε να διακρίνεται αν πρόκειται να επιλεγεί το συγκεκριμένο πεδίο. Επιπλέον, προσθέσαμε κι ένα κουμπί το οποίο μας μεταφέρει στην προηγούμενη σελίδα από αυτή που βρισκόμαστε, σε περίπτωση που δεν είμαστε στην αρχική. Για την εμφάνιση των αποτελεσμάτων μας αξιοποιήσαμε πίνακες, πράγμα σχεδόν αυτονόητο. Η προσπάθειά μας έγκειται στην ομοιομορφία και το ευπαρουσίαστο της εφαρμογής. Κάτι τέτοιο θεωρήσαμε πως θα βοηθούσε το χρήστη ακόμα πιο πολύ στην εύκολη χρήση της.

<u>Προβλήματα</u>

Κατά τη διάρκεια εκπόνησης της εργασίας ήρθαμε αντιμέτωποι με διάφορα προβλήματα. Αυτά αφορούσαν στην υλοποίηση της βάσης αλλά και στη σύνδεσή της εφαρμογής μας με το site. Πιο συγκεκριμένα:

- Ως προς την υλοποίηση της βάσης:
  - Η δημιουργία της βάσης εμφάνισε πρόβλημα όταν προσπαθήσαμε να «δέσουμε» τους πίνακες μεταξύ τους μέσω των κλειδιών. Δηλαδή, δεν μπορούσαμε να δώσουμε με σωστό τρόπο κλειδιά με πολλά γνωρίσματα. Για να το αντιμετωπίσουμε λάβαμε τον κώδικα SQL της βάσης γράψαμε την εντολή που θέλαμε να υλοποιηθεί σε κάθε περίπτωση όσον αφορά στα κλειδιά και κάναμε ξανά εισαγωγή των κώδικα ως βάση πλέον.

- Ως προς τη σύνδεση της εφαρμογής με τη βάση μας:
  - Ενώ αρχικά φάνηκε να πηγαίνουν όλα καλά και να λειτουργούν όπως πρέπει, σε μία από τις εκδόσεις του προγράμματος που χρησιμοποιήσαμε, την πιο καινούρια από αυτές, δίναμε σε κώδικα sql εντολή να υλοποιήσει ένα update και το localhost εμφάνιζε σφάλμα. Την ίδια στιγμή, το ίδιο αίτημα θέταμε σε μία πιο παλιά έκδοση του wamp και υλοποιούνταν χωρίς κανένα πρόβλημα.
  - Άλλο πρόβλημα με το οποίο ήρθαμε αντιμέτωποι ήταν πως ενώ ανανεώναμε τα αρχεία στο φάκελο του wamp στον υπολογιστή μας, το site δεν μπορούσε να αντιληφθεί τις αλλαγές. Στη συγκεκριμένη περίπτωση, ωστόσο, ενώ ο Firefox δεν αντιλαμβανόταν τη διαφορά, ο Chrome μπορούσε να «δει» τα ενημερωμένα αρχεία και να μας εμφανίσει μία σωστή υλοποίηση του κώδικα των σελίδων.

Σχεδιασμός Βάσης

Περιορισμοί:

Για τον ορθότερο σχεδιασμό της βάσης και τη σωστή λειτουργία της χρειάστηκε να θέσουμε κάποιους περιορισμούς όσον αφορά είτε στο τι εισάγει ο χρήστης, είτε στις τιμές που μπορούν να πάρουν τα πεδία στις φόρμες που συμπληρώνουμε στην εφαρμογή μας.

Borrower:

- BId: Primary Key ο κωδικός του δανειζόμενου.

Commitment:

- LId, BId, DateofRequest: Ο συνδυασμός αυτών αποτελεί το Primary Key της σχέσης
- LId: Foreign Key που αναφέρεται στη σχέση Lender. Συγκεκριμένα, αναφέρεται στον δανειστή που έκανε τη δέσμευση.
- Bid, DateofRequest: Ο συνδυασμός τους αποτελεί Foreign Key που αναφέρεται στη σχέση LoanRequest και δείχνει την αίτηση δανείου για την οποία έγινε η δέσμευση.

Deadline:

- Id, DateofAgreement: Ο συνδυασμός τους αποτελεί Primary Key της σχέσης.
- Id: Foreign Key που αναφέρεται στη σχέση Loan και μας πληροφορεί για το δάνειο στο οποίο ανήκει η προθεσμία.

Intermediary:

- MId: Primary Key ο κωδικός του μεσάζοντα.

Lender:

- LId: Primary Key ο κωδικός του δανειστή.

Loan:

- Id: Primary Key, ο κωδικός του δανείου
- MId: Foreign Key που αναφέρεται στη σχέση Intermediary και συγκεκριμένα στον μεσάζονται για το συγκεκριμένο δάνειο.
- BId, DateofRequest: Ο συνδυασμός τους αποτελεί Foreign Key που αναφέρεται στη σχέση LoanRequest και δείχνει σε ποια αίτηση δανείου αντιστοιχεί το δάνειο.

LoanRequest:

- BId, DateofRequest: Ο συνδυασμός τους αποτελεί Primary Key της σχέσης.
- BId: Foreign Key που αναφέρεται στη σχέση Borrower και μας πληροφορεί για τον δανειζόμενο που έκανε την αίτηση.

Repayment:

- Id, DateofPayment: Ο συνδυασμός τους αποτελεί Primary Key της σχέσης.
- Id: Foreign Key που αναφέρεται στη σχέση Loan και μας δείχνει το δάνειο στο οποίο ανήκει η αποπληρωμή.

Trust:

- BId, LId: Ο συνδυασμός τους αποτελεί Primary Key της σχέσης.

- BId: Foreign Key που αναφέρεται στη σχέση Borrower και μας πληροφορεί για τον δανειζόμενο στον οποίο αναφέρεται ο δείκτης εμπιστοσύνης.
- LId: Foreign Key που αναφέρεται στη σχέση Lender και μας πληροφορεί για τον δανειστή που δίνει το δείκτη εμπιστοσύνης.

Όλα τα γνωρίσματα είναι NOT NULL. Εξαίρεση αποτελούν τα Deadline, Amount, PaybackPeriod, Description και Percentage της σχέσης LoanRequest προκειμένου να λειτουργήσει σ ’αυτή ένα trigger που ορίσαμε καθώς και το Town της σχέσης Borrower για την περίπτωση που δημιουργηθεί row μέσω της ενημερώσιμης όψης που θα περιγράψουμε αργότερα.

Στις σχέσεις Borrower, Lender, Intermediary και Loan τα αντίστοιχα primary keys τους BId, LId, MId και Id είναι AUTO_INCREMENT. Έτσι η βάση επιλέγει τα παραπάνω για το χρήστη αντί να ψάχνει ο ίδιος να βρει κάποιο κλειδί που είναι διαθέσιμο.

Όλοι οι αριθμοί που χρησιμοποιούνται δε μπορούν να λάβουν αρνητική τιμή (min="0"). Το ποσοστό εμπιστοσύνης Percentage στη σχέση Trust και το επιτόκιο αποπληρωμής Percentage στη σχέση LoanRequest δε μπορούν να λάβουν τιμές μεγαλύτερες του 100 (max="100"). Επιπλέον, το επιτόκιο αποπληρωμής μπορεί να λάβει και δεκαδικές τιμές (step="any"). Τέλος, στην ενημερώσιμη όψη το PostalCode πρέπει να έχει τιμές μεγαλύτερες του 15100 (min="15101").

Για τα Foreign Keys χρησιμοποιούμε ON DELETE CASCADE. Έτσι, εάν διαγραφεί μια καταχώρηση σε πίνακα του οποίου κάποιο πεδίο αποτελεί Foreign Key σε άλλους, θα διαγραφούν και οι καταχωρήσεις των πινάκων αυτών που είχαν ως Foreign Key το πεδίο που χάθηκε.

Ευρετήρια:

Για κάθε σχέση έχουμε B-Tree ευρετήριο στο Primary Key, που μπορεί να είναι και Foreign Key που αναφέρεται σε άλλη σχέση, να χρησιμοποιείται σαν Foreign Key από άλλη σχέση ή συνδυασμός αυτών.

Οι προκαθορισμένες ερωτήσεις για τις οποίες έχουμε προνοήσει στη βάση μας έχουν ως εξής:

- *Insert:* Ο χρήστης μπορεί να εισάγει μία πλειάδα σε έναν από τους πίνακες που έχουν οριστεί. Η βάση φροντίζει για τη δημιουργία της αντίστοιχης πλειάδας σε όποιον άλλον πίνακα χρειάζεται.
- *Update:* Ο χρήστης μπορεί να ενημερώσει τη βάση του, να αλλάξει ορισμένα από τα δεδομένα των πινάκων χωρίς περιττές κινήσεις.
- *Delete:* Ο χρήστης μπορεί να διαγράψει κάποια πλειάδα από κάποιον από τους πίνακες που επεξεργάζεται. Η βάση και πάλι θα φροντίσει να καλύψει το κενό που δημιουργήθηκε και να διαγράψει τις αντίστοιχες συνδέσεις.
- *DB Info:* Αυτή η επιλογή δίνει στο χρήστη πρόσβαση στα δεδομένα της βάσης, δηλαδή, τους πίνακες με όλα τα στοιχεία για κάθε πίνακα, δηλαδή, κάθε σχέση.
- *Views:* Πρόκειται για έναν τρόπο προβολής των πινάκων που επιλέγουμε. Εδώ έχουμε Updateable και Non-Updateable όψη.

  Updateable View:

```
CREATE ALGORITHM=UNDEFINED
DEFINER=`root`@`localhost` SQL SECURITY DEFINER
VIEW
`thea` AS select
`borrower`.`BId` AS `borrowerid`
,`borrower`.`Name` AS `borrowername`
,`borrower`.`StreetName` AS `streetname`
,`borrower`.`StreetNumber` AS `streetnumber`
,`borrower`.`PoastalCode` AS `postalcode`
from `borrower`
where `borrower`.`PoastalCode`>15100
order by `borrower`.`Name`;
```

  Στην πρώτη περίπτωση, για κάθε ενδεχόμενο στο οποίο υποθέτουμε πως κάνουμε κάποια αλλαγή στην όψη, τότε

ενημερώνεται και η βάση, δηλαδή είναι σαν να κάνουμε αλλαγή στη βάση μέσω ενός χρήστη κι όχι μέσω του ανώτερου που διαχειρίζεται τη βάση. Η πρώτη, αυτή, όψη μας επιστρέφει τον κωδικό δανειζόμενου, όνομα δανειζόμενου, όνομα οδού, αριθμό οδού και ταχυδρομικό κώδικα των δανειζόμενων που μένουν σε περιοχή με ταχυδρομικό κώδικα >15100, με αλφαβητική σειρά ως προς τα ονόματα των δανειζόμενων. Είναι ενημερώσιμη γιατί δε χρησιμοποιεί συναθροιστικές συναρτήσεις και περιέχει στοιχεία από έναν μόνο πίνακα (borrower).

Non-Updateable View:

```
CREATEALGORITHM=UNDEFINED DEFINER =
`root`@`localhost` SQL SECURITY DEFINER VIEW
`v` AS select `l`.`Name` AS `lendername`,
`c`.`LId` AS `lenderid`,
sum(`c`.`Amount`) AS `TotalAmount`
from (`lender` `l` join `commitment` `c`)
where (`l`.`LId` = `c`.`LId`)
group by `c`.`LId`
having (sum(`c`.`Amount`) > 10000)
order by sum(`c`.`Amount`) desc;
```

Η περίπτωση αυτή μας δίνει απλά την εμφάνιση των πινάκων που επιλέγουμε, ενώ οποιαδήποτε αλλαγή σε αυτή δεν ενημερώνει τη βάση, δηλαδή παραμένει στη συγκεκριμένη στιγμή της εμφάνισης που την επιλέγουμε. Επομένως, αυτή η όψη επιστρέφει το όνομα, κωδικό και συνολικό ποσό των δανειστών με τα μεγαλύτερα ποσά >10000 που είναι διατεθειμένοι να δανείσουν, με φθίνουσα σειρά των συνολικών αυτών ποσών. Είναι μη ενημερώσιμη γιατί έχει στοιχεία από δύο πίνακες (lender, commitment) και επίσης χρησιμοποιεί τη συναθροιστική συνάρτηση sum.

- *Queries:* Σε αυτήν την επιλογή δίνουμε στο χρήστη τη δυνατότητα να θέσει ορισμένες ερωτήσεις στη βάση του. Οι ερωτήσεις αυτές είναι προκαθορισμένες από τον προγραμματιστή και πρόκειται για ένωση πινάκων, εύρεση μέγιστου, μέσου όρου και άλλων. Συγκεκριμένα:

1. Join1

```
SELECT *
FROM borrower
JOIN loanrequest ON
borrower.bid= loanrequest.bid
WHERE
loanrequest.percentage>10;
```

Το παραπάνω ερώτημα θα μας εμφανίσει τους δανειζόμενους μαζί με τις αιτήσεις που έχουν κάνει για δάνεια.

2. Join2

```
SELECT
lender.lid As "lender's id",
lender.name As "lender's name",
commitment.bid As "borrower's id",
commitment.dateofrequest,
commitment.amount
FROM
Lender
JOIN commitment ON
lender.lid= commitment.lid
WHERE
commitment.amount>1000;
```

Το ερώτημα αυτό ενώνει δανειστές με δεσμεύσεις, ενώ θα μας εμφανίσει μόνο συγκεκριμένα πεδία.

3. AVG

```
SELECT ROUND(AVG(amount),2) FROM repayment;
```

Αυτό το ερώτημα μας δίνει ως απάντηση το μέσο ποσό ανά δόση, λαμβάνοντα υπ' όψιν όλες τις δόσεις από κάθε δανειζόμενο.

4. Sum,GroupBy,OrderBy,Limit,Nested query

```
SELECT name
FROM lender
WHERE  lid=(SELECT lid FROM commitment c
        GROUP BY c.lid
        ORDER BY Sum(amount)
    DESC limit 1);
```

Αυτό το ερώτημα θα μας εμφανίζει το δανειστή που έχει δανείσει τα περισσότερα συνολικά χρήματα.

5. Group By

```
SELECT bid, COUNT(*)
FROM loanrequest
GROUP BY bid;
```

Θα μας δείξει πόσες αιτήσεις έχει κάνει ο κάθε χρήστης.

6. OrderBy

```
SELECT * FROM borrower
ORDER by name desc
limit 8;
```

Αυτό το ερώτημα ως απάντηση θα μας εμφανίσει τους δανειζόμενους ταξινομημένους με φθίνουσα σειρά κατά όνομα και θα μας δείξει τους πρώτους 8.

7. GroupBy having

```
SELECT lender.Name,COUNT(*) as
"NumberofCommitments" FROM (commitment
INNER JOIN lender ON
  commitment.lid=lender.lid)
GROUP BY lender.name
having count(*)>=2;
```

Σε αυτήν την περίπτωση θα εμφανιστεί πίνακας με τους δανειστές που έχουν συμφωνήσει σε τουλάχιστον 2 δεσμεύσεις.

8. Nested query

```
select Name from borrower b
   where exists
      (select * from loan d
      where b.bid=d.bid);
```

Θα μας εμφανίσει τους δανειζόμενους για τους οποίους έχει εγκριθεί τουλάχιστον ένα δάνειο.

9. Like

```
SELECT * FROM lender
WHERE name LIKE '%papak_nstantino%';
```

Αυτό το ερώτημα θα μας εμφανίσει τις καταχωρήσεις που περιλαμβάνουν τους χαρακτήρες «Παπακ%νσταντίνο%» με διαφορετικές ορθογραφίες στις θέσεις των «%».

10. Union

```
(select Name from intermediary, loan
 where intermediary.mid=loan.mid and
Dateofapproval='2010-12-05')
union
(select Name from intermediary, loan
 where intermediary.mid=loan.mid and
Dateofrequest='2012-11-19');
```

Θα μας εμφανίσει τα ονόματα των μεσαζόντων που έχουν εμπλακεί σε δάνειο μία συγκεκριμένη ημερομηνία αίτησης ή έγκρισης.

Επιπλέον, στο σχεδιασμό της βάσης μας, μας ζητήθηκε να προνοήσουμε για 2 triggers της επιλογής μας. Αυτό σημαίνει, πως όταν συμβαίνει ένα συγκεκριμένο γεγονός, τότε αυτά πυροδοτούνται και προστατεύουν τα δεδομένα μας και τη σωστή λειτουργία της βάσης μας. Αυτά επεξηγούνται παρακάτω:

- Trigger1:

```
CREATE TRIGGER `trigger1` AFTER INSERT ON
`commitment`
FOR EACH ROW insert into loan
(dateofapproval,mid,bid,dateofrequest)
    select CURDATE(),(SELECT mid FROM intermediary
ORDER BY RAND() LIMIT 1),l.bid,l.dateofrequest from
loanrequest l
    where l.amount <=
    (select sum(d.amount) from commitment d
     where l.bid=d.bid and
l.dateofrequest=d.dateofrequest
     group by d.bid,d.dateofrequest )
     and not exists (select 1
                     from loan
                     where bid=l.bid
                    and
     dateofrequest=l.dateofrequest);
```

Μέσω αυτού του trigger ελέγχεται ο πίνακας commitment κάθε φορά που γίνεται εισαγωγή για κάποια αίτηση δανείου, ώστε άμα συμπληρωθεί το ποσό προσθέτει την καταχώρηση στον πίνακα loan. Το id του πίνακα loan αυξάνει αυτόματα και ως ημερομηνίας έγκρισης τίθεται η ημερομηνία κατά την οποία συνέβη αυτό το γεγονός. Ως

μεσάζοντας επιλέγεται κάποιος τυχαίος από τους καταχωρημένους στη βάση μας και σε περίπτωση που κάποιο δάνειο υπάρχει ήδη, τότε δεν ξαναμπαίνει στον πίνακά μας.

- Trigger2:

```
CREATE TRIGGER `trigger2` AFTER INSERT ON `borrower`
FOR EACH ROW INSERT INTO loanrequest VALUES
 (new.bid,CURDATE(),NULL,NULL,NULL,NULL,NULL);
```

Στην περίπτωση αυτή κάθε φορά που προστίθεται ένας δανειζόμενος στον αντίστοιχο πίνακα δημιουργείται αυτόματα μία αίτηση νέου δανείου στην οποία συμπληρώνονται ο αύξοντας αριθμός του δανειζόμενου και η ημερομηνία αίτησης. Τα υπόλοιπα πεδία τίθενται NULL.

Τέλος, τα DLL που χρησιμοποιήθηκαν για τη δημιουργία των πινάκων βάσης είναι τα εξής:

```
CREATE TABLE IF NOT EXISTS `borrower` (
  `BId` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) NOT NULL,
  `Town` varchar(30),
  `StreetName` varchar(30) NOT NULL,
  `StreetNumber` int(11) NOT NULL,
  `PoastalCode` int(11) NOT NULL,
  PRIMARY KEY (`BId`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=11
;

INSERT INTO `borrower` (`BId`, `Name`, `Town`,
`StreetName`, `StreetNumber`, `PoastalCode`) VALUES
(1, 'KWSTAS  KOBITSAKIS', 'ELEYSINA', 'MANDILARA', 42,
12345),
(2, 'GIORGOS XYDAS', 'SAMOS', 'VYZANTIOU', 95, 12679),
(3, 'MANOLIS MANDILARAS', 'MYTILINH', 'KOPEGXAGHS', 78,
12489),
(4, 'KATERINA KORONAIOU', 'ANTIPAROS', 'SOULIOY', 176,
13497),
(5, 'ELSA TOYRNAVITOY', 'KRHTH', 'IEROU LOXOU', 18,
18946),
(6, 'GIANNIS SIGLIDIS', 'KEFALONIA', 'SPARTAKOY', 5,
18645),
```

```
(7, 'FTWXOS SYGGENIS', 'NAXOS', 'ETHNIKIS ANTISTASEWS',
79, 17264),
(8, 'GIORGOS SINTOSIS', 'PAROS', 'SKRA', 4, 18678),
(9, 'KOSTAS KOULOGLOU', 'MYKONOS', 'KOLOKOTRONI', 3,
14563),
(10, 'ANTREI ZAXAROF', 'KATAPOLA', 'SKARAMAGKA', 89,
13245);



CREATE TABLE IF NOT EXISTS `commitment` (
  `LId` int(11) NOT NULL,
  `Bid` int(11) NOT NULL,
  `DateofRequest` date NOT NULL,
  `Amount` int(11) NOT NULL,
  PRIMARY KEY (`LId`,`Bid`,`DateofRequest`),
  KEY `commitment_ibfk_2` (`Bid`,`DateofRequest`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



INSERT INTO `commitment` (`LId`, `Bid`, `DateofRequest`,
`Amount`) VALUES
(1, 2, '2010-11-05', 22000),
(1, 8, '2015-05-02', 10000),
(2, 1, '2011-08-09', 2000),
(2, 9, '2012-11-19', 1000),
(3, 1, '2010-07-12', 30000),
(4, 1, '2010-07-12', 70000),
(4, 9, '2014-09-27', 3000),
(5, 2, '2010-11-05', 20000),
(6, 2, '2015-03-29', 1000),
(7, 3, '2014-05-22', 3900),
(8, 4, '2013-05-16', 2000),
(8, 10, '2013-03-12', 1000),
(9, 8, '2015-05-02', 22000),
(9, 9, '2012-11-19', 1000),
(10, 9, '2012-11-19', 1000);

CREATE TABLE IF NOT EXISTS `deadline` (
  `Id` int(11) NOT NULL,
  `DateofAgreement` date NOT NULL,
  `Deadline` date NOT NULL,
  PRIMARY KEY (`Id`,`DateofAgreement`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



INSERT INTO `deadline` (`Id`, `DateofAgreement`,
`Deadline`) VALUES
(1, '2010-08-12', '2016-02-19'),
(2, '2010-12-05', '2015-12-09'),
```

```
(2, '2015-12-05', '2020-12-09'),
(3, '2014-06-22', '2015-12-19'),
(3, '2015-11-22', '2018-12-19'),
(4, '2015-06-02', '2022-11-21'),
(5, '2012-12-19', '2020-12-19');
CREATE TABLE IF NOT EXISTS `intermediary` (
  `MId` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) NOT NULL,
  `Town` varchar(30) NOT NULL,
  `StreetName` varchar(30) NOT NULL,
  `StreetNumber` int(11) NOT NULL,
  `PoastalCode` int(11) NOT NULL,
  PRIMARY KEY (`MId`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;



INSERT INTO `intermediary` (`MId`, `Name`, `Town`,
`StreetName`, `StreetNumber`, `PoastalCode`) VALUES
(1, 'IASIS', 'ATHENS', 'RODOY', 56, 10524),
(2, 'PRAKSIS', 'ATHENS', 'TRIKOYPH', 80, 14275),
(3, 'AIGIALI', 'AMORGOS', 'THYRAS', 5, 18923),
(4, 'YOUTHFULLY', 'THESSALONIKH', 'PANORAMA', 82, 12398),
(5, 'OHE', 'NEW YORK', '113TH STREET', 199, 10256);



CREATE TABLE IF NOT EXISTS `lender` (
  `LId` int(11) NOT NULL AUTO_INCREMENT,
  `Name` varchar(50) NOT NULL,
  `Town` varchar(30) NOT NULL,
  `StreetName` varchar(30) NOT NULL,
  `StreetNumber` int(11) NOT NULL,
  `PoastalCode` int(11) NOT NULL,
  PRIMARY KEY (`LId`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=11
;

INSERT INTO `lender` (`LId`, `Name`, `Town`, `StreetName`,
`StreetNumber`, `PoastalCode`) VALUES
(1, 'GIORGOS PAPAKONSTANTINOU', 'MAROUSSI',
'THEMISTOKLEOUS', 31, 15124),
(2, 'TROIKA', 'FLWRINA', 'AVEROF', 13, 12356),
(3, 'VOLFGANG SOIMPLE', 'KOZANH', 'KOLETTH', 59, 14598),
(4, 'VASILEIOS MPATSHS', 'BERGAMO', 'MARATHONOMAXON', 45,
14637),
(5, 'INTERNATIONAL MOMENTARY FUND', 'NEW YORK',
'TRIPOLEOS', 61, 16873),
(6, 'VASILIS PAPAKWNSTANTINOU', 'MADRID',
'MARATHONODROMON', 27, 17592),
(7, 'THANASIS PAPAKWNSTANTINOY', 'KORNOUALH', 'ELIKONOS',
185, 14838),
```

```
(8, 'GIORGOS TIRININIS', 'AIKATERINH', 'KEAS', 49, 18938),
(9, 'KYRIAZHS NIKOS', 'ATHENS', 'DRAGATSANIOU', 48,
18927),
(10, 'VAGGELIS EUSTHATHIOU', 'THESSALONIKH', 'TOUMPAS',
13, 13574);



CREATE TABLE IF NOT EXISTS `loan` (
  `Id` int(11) NOT NULL AUTO_INCREMENT,
  `DateofApproval` date NOT NULL,
  `MId` int(11) NOT NULL,
  `BId` int(11) NOT NULL,
  `DateofRequest` date NOT NULL,
  PRIMARY KEY (`Id`),
  KEY `MId` (`MId`),
  KEY `loan_ibfk_2` (`BId`,`DateofRequest`)
) ENGINE=InnoDB  DEFAULT CHARSET=latin1 AUTO_INCREMENT=6 ;



INSERT INTO `loan` (`Id`, `DateofApproval`, `MId`, `BId`,
`DateofRequest`) VALUES
(1, '2010-08-12', 2, 1, '2010-07-12'),
(2, '2010-12-05', 3, 2, '2010-11-05'),
(3, '2014-06-22', 1, 3, '2014-05-22'),
(4, '2015-06-02', 4, 8, '2015-05-02'),
(5, '2012-12-19', 5, 9, '2012-11-19');



CREATE TABLE IF NOT EXISTS `loanrequest` (
  `BId` int(11) NOT NULL,
  `DateofRequest` date NOT NULL,
  `Deadline` date DEFAULT NULL,
  `Amount` int(11) DEFAULT NULL,
  `PaybackPeriod` varchar(100) DEFAULT NULL,
  `Description` text,
  `Percentage` float DEFAULT NULL,
  PRIMARY KEY (`BId`,`DateofRequest`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



INSERT INTO `loanrequest` (`BId`, `DateofRequest`,
`Deadline`, `Amount`, `PaybackPeriod`, `Description`,
`Percentage`) VALUES
(1, '2010-07-12', '2020-07-12', 100000, '10', 'investment
in new machinery', 10.5),
(1, '2011-08-09', '2023-08-09', 16000, '12', 'buying a
car', 7.5),
```

```
(2, '2010-11-05', '2025-11-05', 42000, '15', 'buying a
house', 16.3),
(2, '2015-03-29', '2017-03-29', 3600, '2', 'repairs',
13.72),
(3, '2014-05-22', '2017-05-22', 3900, '3', 'investment in
Rnd department', 19.7),
(4, '2013-05-16', '2018-05-16', 8200, '5', 'investment in
new offices', 3.72),
(5, '2013-05-16', '2017-05-16', 25000, '4', 'in order to
expand production circle', 1.16),
(6, '2012-03-13', '2023-03-13', 10000, '11', 'initial
bugdet for startup', 17.43),
(7, '2013-11-19', '2028-11-19', 114000, '15', 'investment
in warehouse', 15.23),
(8, '2011-02-19', '2023-02-19', 15000, '12', 'buying a
car', 3.2),
(8, '2015-05-02', '2045-05-02', 32000, '30', 'need for
lyquidity', 9.7),
(9, '2012-11-19', '2017-11-19', 3000, '5', 'vacation
loan', 13),
(9, '2014-09-27', '2021-09-27', 5000, '7', 'investment in
new equipment', 16.16),
(10, '2013-03-12', '2038-03-12', 50000, '25', 'buying a
house', 6.3),
(10, '2015-12-17', '2047-12-17', 70000, '32', 'in order to
renovate equipment', 12.42);



CREATE TABLE IF NOT EXISTS `repayment` (
  `Id` int(11) NOT NULL,
  `DateofPayment` date NOT NULL,
  `Amount` int(11) NOT NULL,
  PRIMARY KEY (`Id`,`DateofPayment`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
INSERT INTO `repayment` (`Id`, `DateofPayment`, `Amount`)
VALUES
(1, '2011-08-12', 1000),
(2, '2011-12-05', 2000),
(2, '2012-12-05', 500),
(2, '2014-12-05', 1000),
(3, '2014-07-22', 1500),
(3, '2014-08-22', 900),
(3, '2014-09-22', 900),
(4, '2015-07-02', 1100),
(4, '2015-08-02', 1100),
(5, '2013-12-19', 3500);



CREATE TABLE IF NOT EXISTS `trust` (
  `BId` int(11) NOT NULL,
```

```
  `LId` int(11) NOT NULL,
  `Percentage` int(11) NOT NULL,
  PRIMARY KEY (`BId`,`LId`),
  KEY `trust_ibfk_2` (`LId`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;



INSERT INTO `trust` (`BId`, `LId`, `Percentage`) VALUES
(1, 5, 50),
(1, 6, 46),
(2, 2, 32),
(3, 10, 10),
(4, 4, 87),
(4, 9, 65),
(5, 10, 30),
(6, 1, 5),
(6, 10, 48),
(7, 9, 98),
(8, 7, 56),
(9, 3, 100),
(9, 5, 80),
(9, 6, 20),
(10, 7, 79);



ALTER TABLE `commitment`
ADD CONSTRAINT `commitment_ibfk_2` FOREIGN KEY
(`Bid`, `DateofRequest`) REFERENCES `loanrequest`
(`BId`, `DateofRequest`) ON DELETE CASCADE ON UPDATE
CASCADE,
ADD CONSTRAINT `commitment_ibfk_1` FOREIGN KEY (`LId`)
REFERENCES `lender` (`LId`) ON DELETE CASCADE ON UPDATE
CASCADE;


ALTER TABLE `deadline`
ADD CONSTRAINT `deadline_ibfk_1` FOREIGN KEY (`Id`)
REFERENCES `loan` (`Id`) ON DELETE CASCADE ON UPDATE
CASCADE;


ALTER TABLE `loan`
ADD CONSTRAINT `loan_ibfk_2` FOREIGN KEY
(`BId`,`DateofRequest`) REFERENCES `loanrequest` (`BId`,
`DateofRequest`)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `loan_ibfk_1` FOREIGN KEY (`MId`)
REFERENCES `intermediary` (`MId`)
ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `loanrequest`
ADD CONSTRAINT `loanrequest_ibfk_1` FOREIGN KEY (`BId`)
REFERENCES `borrower` (`BId`)
ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `repayment`
ADD CONSTRAINT `repayment_ibfk_1` FOREIGN KEY (`Id`)
REFERENCES `loan` (`Id`)
ON DELETE CASCADE ON UPDATE CASCADE;


ALTER TABLE `trust`
ADD CONSTRAINT `trust_ibfk_2` FOREIGN KEY (`LId`)
REFERENCES `lender` (`LId`)
ON DELETE CASCADE ON UPDATE CASCADE,
ADD CONSTRAINT `trust_ibfk_1` FOREIGN KEY (`BId`)
REFERENCES `borrower` (`BId`)
ON DELETE CASCADE ON UPDATE CASCADE;
```