ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

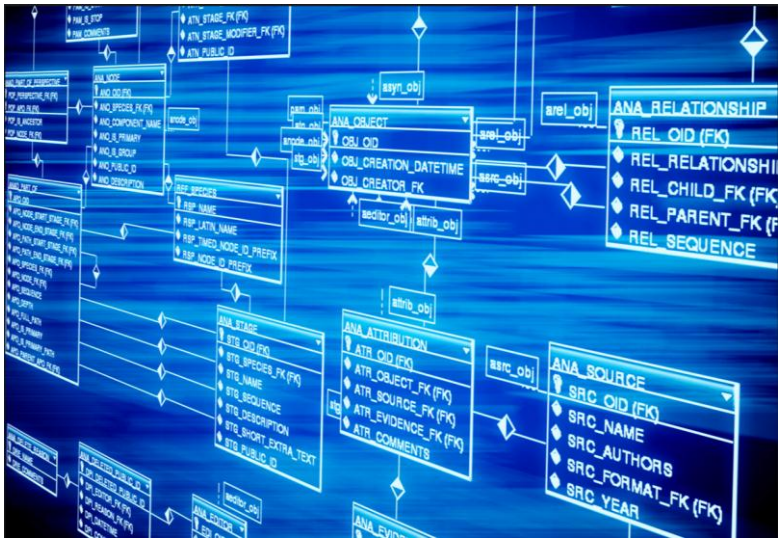
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΜΑΘΗΜΑ : ***ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ (ΡΟΗ Λ)***

ΕΞΑΜΗΝΟ : ***7ο***

PROJECT : Σχεδιασμός Βάσης Δεδομένων

«MICRO-LOANS»

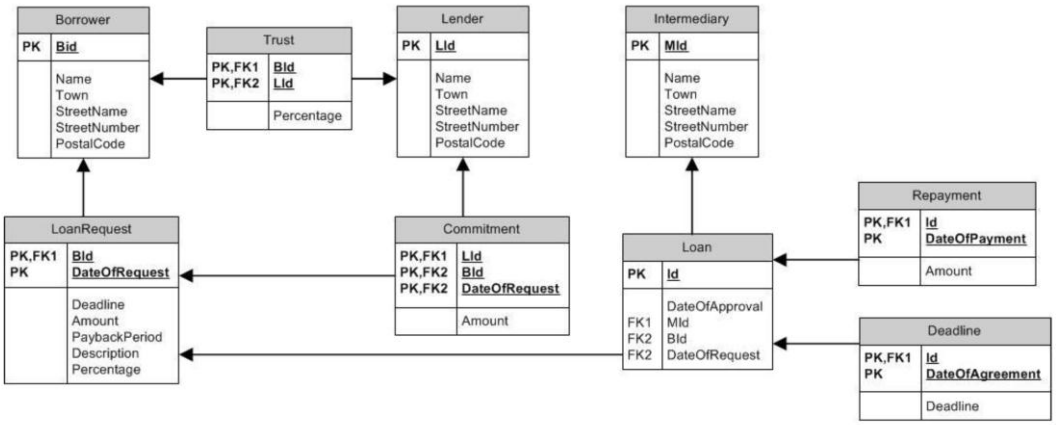


ΟΜΑΔΑ (3 ατόμων):

1. **Γιαννόπουλος Αναστάσιος** (03112176)
2. **Αθανασίου Νικόλαος** (03112074)
3. **Καραμπλιάς Φώτιος** (03112004)

ΕΙΣΑΓΩΓΗ:

Η παρούσα εργασία αφορά στην ανάπτυξη σε σχεσιακό σύστημα μιας βάσης δεδομένων που αφορά ΜΙΚΡΟΔΑΝΕΙΑ**.** Το σχεσιακό μοντέλο (όπως χρησιμοποιήθηκε απ' την λύση της 1ης άσκησης) έχει ως εξής:



ΧΡΗΣΙΜΑ ΕΡΓΑΛΕΙΑ:

Για την υλοποίηση της βάσης έγινε χρήση:

- XAMPP: το οποίο παρείχε τον APACHE (Web Server) και MySql Database Management System.
- PHP: Scripting Language για τη σύνδεση της ιστοσελίδας με την DB.
- JAVASCRIPT: Scripting Language για την υλοποίηση βασικών λειτουργειών των πινάκων της DB.
- CSS: για τη μορφοποίηση του Web που δημιουργήσαμε.
- HTML: για την δημιουργία της ιστοσελίδας που αποτελεί μια απεικόνιση μιας DB.

Ουσιαστικά η υλοποίηση του Project έγινε σε μορφή ιστοσελίδας, η οποία περιέχει τα κατάλληλα menu και μορφοποίηση, ώστε να μπορεί η χρήστης χωρίς τεχνικές γνώσεις να χειριστεί τη DB, να προβάλει τα queries και τα views του συστήματος. Με τη βοηθεια των παραπάνω γλωσσών και εργαλείων υλοποιήθηκε ένα περιβάλλον φιλικό προς το χρήστη. Κάθε ιστοσελίδα (αρχεία .php), καθώς και των αντίστοιχων insert, delete και update, έγινε σε ξεχωριστά αρχεία με ίδιο style (αρχεία .css), ώστε ο κώδικας να είναι πιο ευέλικτος και ανεξάρτητος. Επίσης, στη DB προστέθηκε κωδικός, η αφαίρεση του οποίου απ' το κώδικα, αποτρέπει τη σύνδεση για περισσότερη ασφάλεια.

ΠΛΕΟΝΕΚΤΗΜΑΤΑ:

1. Η κατασκευή σε μορφή ιστοσελίδων επιτρέπει πρόσβαση από διάφορους Η/Υ χώρις την εμφάνιση διαφορών σε διαφορετικά Operating Systems. Δηλαδή ο κώδικας εμφανίζει καθολικότητα.
2. Εύκολη και κατανοητή σύνταξη κώδικα λόγω της αλληλουχίας και ευκολίας σύνδεσης της βάσης στη σελίδα (html code) μέσω της php, η οποία είναι υπεύθυνη για τη διαχείρηση της βάσης (λόγω της SQL που συντάσσεται μέσα της).
3. Κατανοητό και εύχρηστο interface για το χρήστη μέσω της χρήσης της ιστοσελίδας.

4. Το περιβάλλον που χρησιμοποιήσαμε παρείχε σημαντικά μηνύματα πολύ κατατοπιστικά για το debugging του project.

ΜΕΙΟΝΕΚΤΗΜΑΤΑ:

1. Η SQL δεν ελέγχει από μόνη της τους περιορισμούς, οπότε θα πρέπει να ελεγχούν απ' τη γλώσσα που υποστηρίζει την ιστοσελίδα (html & javascript).

ΛΕΙΤΟΥΡΓΙΚΕΣ ΑΠΑΙΤΗΣΕΙΣ - ΠΡΟΔΙΑΓΡΑΦΕΣ:

Οι οντότητες και οι σχέσεις της DB μας έχουν τους εξής περιορισμούς:

1. Για κάθε Δανειστή και κάθε Δανειζόμενο κρατάμε το Όνομα και η Διεύθυνση των.
2. Ένα Δάνειο ξεκινά με Αίτηση αυτού που θέλει το δάνειο, η οποία περιέχει πληροφορίες για το πότε πρέπει να δοθεί το δάνειο το αργότερο, το συνολικό ποσό του δανείου, και ποια είναι η περίοδος αποπληρωμής (payback period). Επίσης, χρειάζεται και μια περιγραφή για το πώς θα χρησιμοποιηθεί το δάνειο. Το δάνειο αποπληρώνεται με ένα επιτόκιο που αναφέρεται στην αίτηση ως ποσοστό στο συνολικό ποσό του δανείου.
3. Όταν έχει γίνει το αίτημα για δάνειο, ένας Δανειστής μπορεί να δεσμευτεί ότι θα καλύψει κάποιο μέρος του δανείου.
4. Όταν οι δεσμεύσεις για το δάνειο καλύψουν το συνολικό αιτούμενο ποσό, τότε δίνεται το δάνειο. Προφανώς, εάν δεν γίνουν οι απαραίτητες δεσμεύσεις, τότε το δάνειο ακυρώνεται. Ένας Δανειζόμενος μπορεί να κάνει αιτήματα για περισσότερα του ενός δάνεια, και να έχει πάρει περισσότερα του ενός δάνεια σε μια συγκεκριμένη χρονική στιγμή. Όμως δεν μπορεί να κάνει περισσότερα του ενός αιτήματα την ίδια μέρα.
5. Για την ασφαλή λειτουργία του συστήματος τα Δάνεια αποπληρώνονται μέσω τρίτου («μεσάζοντα» / «intermediary») που συνήθως είναι ένας μη-κερδοσκοπικός οργανισμός (κρατάμε στη βάση Όνομα και Διεύθυνση του τρίτου).
6. Ο Δανειζόμενος επιλέγει πότε θα πληρώσει μια δόση (payment) για την αποπληρωμή του δανείου. Για κάθε δόση καταγράφονται στη βάση δεδομένων το Ποσό (amount) και η Ημερομηνία καταβολής της δόσης (date). Την ίδια μέρα / ημερομηνία δεν μπορεί να πληρωθούν περισσότερες της μιας δόσης ανά δάνειο. Οι Δανειστές μοιράζονται τη Δόση με βάση το ποσοστό συμμετοχής τους στο Δάνειο.
7. Εάν (συνολικά) το Δάνειο δεν αποπληρωθεί στη συμφωνηθείσα ημερομηνία (deadline), μια νέα ημερομηνία συμφωνείται.
8. Κάθε ένας Δανειστής μπορεί να κρατά ένα δείκτη «εμπιστοσύνης» ("trust") για κάθε Δανειζόμενο. Ο δείκτης είναι ένας αριθμός μεταξύ του 0 και του 100 και αντανακλά το ρίσκο του δανείου και τη φερεγγυότητα του δανειζόμενου.

ΣΧΕΔΙΑΣΜΟΣ ΤΗΣ ΒΑΣΗΣ:

Η σχεδίαση της βάσης υλοποιήθηκε με βάση τη προτεινόμενη λύση. Τα αριθμητικά (PK) κάθε πίνακα παράγονται με αυτόματη αύξηση (auto increment), το οποίο διασφαλίζει την ακεραιότητα οντότητας κάθε εγγραφής στον ίδιο πίνακα. Δηλαδή κάθε εγγραφή στο πίνακά της θα έχει μοναδικό κύριο κλειδί από όλες τις υπόλοιπες. Αυτό συμβαίνει για τους εξής πίνακες: Borrower , Lender , Intermediary , Loan. Για τους υπόλοιπους πίνακες, η ακεραιότητα οντότητας εξασφαλίζεται μέσω του κώδικα, λόγω των εξωτερικών κλειδιών των υπόλοιπων πινάκων, τα οποία αναφέρονται στα παραπάνω πρωτεύοντα κλειδιά. Συγκεκριμένα, όταν πάει να εισαχθεί κάποια εγγραφή σ' αυτούς, ελέγχεται αν το κλειδί υπάρχει

ήδη σε πίνακα. Αν υπάρχει εισάγεται στον αντίστοιχο πίνακα, αλλιώς απαγορεύεται η εισαγωγη (με κατάλληλο μήνυμα).

Όσον αφορά την αναφορική ακεραιότητα, εξασφαλίζεται με τον εξής τρόπο: Σε περίπτωση που διαγράφεται κάποια εγγραφή που χρησιμοποιεί ως FK ένα PK μιας άλλης αυτό επιτρέπεται στο χρήστη, ενώ στη περίπτωση που διαγράφεται κάποια εγγραφή που το PK της εμπλέκεται ως FK σε εγγραφή σ' άλλο πίνακα, τότε δεν επιτρέπεται στη χρήστη διαγραφή, αφού τότε δεν έχει νοήμα η ύπαρξη της εγγραφής που χρησιμοποιεί ως FK το PK της πρώτης. Για παράδειγμα, δεν έχει νόημα η ύπαρξη ενός δανείου που χρησιμοποιεί τα PK του Borrower, Lender αν δεν υπάρχουν οι αντίστοιχες εγγραφές του Borrower, Lender στους πίνακές τους.

Επίσης, εξασφαλίζεται και το domain intergrity (ακεραιότητα πεδίου τιμών) με τους κατάλληλους ορισμούς τύπων δεδομένων για κάθε πεδίο κάθε πίνακα στον κώδικα της SQL.

Τέλος, εξασφαλίζεται και η ακεραιότητα user defined, αφού κάθε οντότητα έχει δικά της πεδία, ανεξάρτητου και διαφορετικού τύπου.

Τα ευρετήρια της βάσης είναι όλα πρωτεύοντα (PK κάθε πίνακα), γεγονός που εξασφαλίζεται απ' τον Apache Web Server.

Όλοι οι υπόλοιποι περιορισμοί και παραδοχές που υποστηρίζονται απ' τη βάση αναφέρονται ρητά στις λειτουργικές απαιτήσεις (βλ. παραπάνω).

TRIGGERS:

Πρόκειται για είναι διαδικαστικό κώδικα που εκτελείται αυτόματα σε απάντηση σε ορισμένα γεγονότα σχετικά με ένα συγκεκριμένο πίνακα ή προβολή σε μια βάση δεδομένων. Το trigger χρησιμοποιείται κυρίως για τη διατήρηση της ακεραιότητας των πληροφοριών στη βάση δεδομένων. Για παράδειγμα, όταν διαγράφουμε έναν Lender πρέπει να διαγράφονται ταυτόχρονα και τα αντίστοιχα πεδία Trust, Commitment ή όταν εισάγεται ένας Borrower πρέπει να εισάγεται και κατάλληλη τιμή στο Deadline.

1. Trigger_1:

```
CREATE TRIGGER `LenderDelete` BEFORE DELETE ON `Lender`
      FOR EACH ROW BEGIN
      DELETE FROM `Commitment`
      WHERE `OLD`.`LId`=`Commitment`.`LId`;
      DELETE FROM `Trust`
      WHERE `OLD`.`LId`=`Trust`.`LId`;
END
```

2. Trigger_2:

```
CREATE TRIGGER `DeadLineId` AFTER INSERT ON `Loan`
      FOR EACH ROW
      INSERT INTO `project`.`DeadLine`(`Id`)
      VALUES (`new`.`Id`)");
```

VIEWS:

Στη θεωρία των βάσεων δεδομένων, ένα view είναι το αποτέλεσμα ενός αποθηκευμένου ερωτήματος σχετικά με τα δεδομένα, το οποίο οι χρήστες της βάσης δεδομένων μπορεί να ρωτήσουν ακριβώς όπως θα έκαναν σε μια επίμονη συλλογή δεδομένων. Αυτή η προ-εγκατεστημένη εντολή (ερώτημα) διατηρείται στο λεξικό δεδομένων. Αντίθετα με τα συνηθισμένους πίνακες βάσης σε μια σχεσιακή βάση δεδομένων, η view δεν αποτελεί μέρος του φυσικού σχήματος: όπως ορίζεται ένα αποτέλεσμα, είναι ένας εικονικός

πίνακας που υπολογίζεται ή αντιπαραβάλλει δυναμικά από τα δεδομένα στη βάση δεδομένων όταν ζητείται η πρόσβαση σε αυτή την όψη.

1. View_1 (ενημερώσιμη):

Αν στο σύστημα της βάσης δεδομένων μπορεί να προσδιοριστεί η αντίστροφη χαρτογράφηση από το view schema στο σχήμα των υποκείμενων πινάκων βάσης, τότε η view είναι updatetable. INSERT, UPDATE και DELETE λειτουργίες μπορούν να εκτελεστούν στην δυνατότητα ενημέρωσης των views. Μια ενημερωμένη view γίνεται με το key preservation.

π.χ.Borrowers με αιτούμενο amount > 1000:

```
CREATE VIEW `Update`(Name,DateOfRequest,Percentage) AS
SELECT `Borrower`.`Name`,`LoanRequest`.`DateOfRequest`,`LoanRequest`.`Percentage`
FROM `Borrower`,`LoanRequest`
WHERE `Borrower`.`BId`=`LoanRequest`.`BId` AND `LoanRequest`.`Amount` > 1000
```

2. View_2 (μη-ενημερώσιμη):

Χρησιμοποιεί συνάρτηση της SQL. Αν για παράδειγμα αλλάξω το πεδίο average μεταξύ 3 τιμών, το σύστημα δε ξέρει ποια/ες απ' τις 3 τιμές μεταβλήθηκε, ώστε να μεταβληθεί ο μέσος όρος. π.χ. Μέσος Όρος των amount που ζητήθηκε και αριθμός αιτήσεων που υπέβαλε καθένας Borrower:

```
CREATE VIEW `CountLoans`(Name,Counter,Amount) AS
SELECT `Borrower`.`Name`,COUNT(*),AVG(`LoanRequest`.`Amount`)
FROM `Borrower`,`LoanRequest`
WHERE `Borrower`.`BId`=`LoanRequest`.`BId`
GROUP BY `Borrower`.`BId`
```

DDL ΚΑΤΑΣΚΕΥΗΣ ΤΗΣ ΒΑΣΗΣ:

```
CREATE TABLE project.Borrower (
BId INT(10) NOT NULL AUTO_INCREMENT ,
Name VARCHAR(40) NULL ,
Town VARCHAR(30) NULL ,
StreetName VARCHAR(30) NULL ,
StreetNumber INT(4) UNSIGNED NULL ,
PostalCode INT(5) UNSIGNED NULL ,
PRIMARY KEY (BId));
```

```
CREATE TABLE project.Lender (
LId INT(10) NOT NULL AUTO_INCREMENT,
Name VARCHAR(40) NULL ,
Town VARCHAR(30) NULL ,
StreetName VARCHAR(30) NULL ,
StreetNumber INT(4) NULL ,
PostalCode INT(5) NULL ,
```

```
PRIMARY KEY (LId));
```

```
CREATE TABLE project.Intermediary (
MId INT(10) NOT NULL AUTO_INCREMENT ,
Name VARCHAR(40) NULL ,
Town VARCHAR(30) NULL ,
StreetName VARCHAR(30) NULL ,
StreetNumber INT(4) NULL ,
PostalCode INT(5) NULL ,
PRIMARY KEY (MId));
```

```
CREATE TABLE project.Trust (
BId INT(10) NOT NULL ,
LId INT(10) NOT NULL ,
Percentage INT(3) NULL ,
FOREIGN KEY (BId) REFERENCES Borrower(BId),
FOREIGN KEY (LId) REFERENCES Lender(LId),
PRIMARY KEY (BId,LId));
```

```
CREATE TABLE project.LoanRequest (
BId INT(10) NOT NULL ,
DateOfRequest DATE NOT NULL ,
DeadLine DATE NULL ,
Amount INT(20) NULL ,
PaybackPeriod DATE NULL ,
Description TEXT(100) NULL ,
Percentage DECIMAL(3,2) NULL ,
FOREIGN KEY (BId) REFERENCES Borrower(BId),
PRIMARY KEY (BId,DateOfRequest));
```

```
CREATE TABLE project.Commitment (
BId INT(10) NOT NULL ,
LId INT(10) NOT NULL ,
DateOfRequest DATE NOT NULL ,
Amount DOUBLE(20,2) NULL ,
FOREIGN KEY (BId,DateOfRequest) REFERENCES
LoanRequest(BId,DateOfRequest),
FOREIGN KEY (LId) REFERENCES Lender(LId),
PRIMARY KEY (LId,BId,DateOfRequest));
```

```
CREATE TABLE project.Loan (
Id INT(10) NOT NULL AUTO_INCREMENT,
DateOfApproval DATE NOT NULL ,
MId INT(10) NOT NULL ,
BId INT(10) NOT NULL ,
DateOfRequest DATE NOT NULL ,
FOREIGN KEY (BId,DateOfRequest) REFERENCES
LoanRequest(BId,DateOfRequest),
```

```
FOREIGN KEY (MId) REFERENCES
Intermediary(MId),
PRIMARY KEY (Id));
```

```
CREATE TABLE project.Repayment (
Id INT(10) NOT NULL ,
DateOfPayment DATE NOT NULL ,
Amount DOUBLE(20,2) NULL ,
FOREIGN KEY (Id) REFERENCES Loan(Id),
PRIMARY KEY (Id,DateOfPayment));
```

```
CREATE TABLE project.DeadLine (
Id INT(10) NOT NULL ,
DateOfAgreement DATE NOT NULL ,
DeadLine DATE NULL ,
FOREIGN KEY (Id) REFERENCES Loan(Id),
PRIMARY KEY (Id,DateOfAgreement));
```

SQL QUERIES:

Query 1:

Απλό ερώτημα που διαλέγει τους lenders που βρίσκονται στην Αθήνα:

```
SELECT Borrower.BId,Borrower.Name
FROM Borrower
WHERE Borrower.Town='Athens'
```

Query 2:

Ταξινόμηση με βάση trust(σε φθίνουσα λόγω ενδιαφέροντος εμπιστοσύνης):

```
SELECT Borrower.Name AS Bname , Trust.Percentage AS TP, Lender.Name AS LN
FROM Borrower, Trust , Lender
WHERE Borrower.BId=Trust.BId AND Trust.LId = Lender.LId
ORDER BY Trust.Percentage DESC
```

Query 3:

Ταξινόμηση με βάση αιτούμενο ποσό:

```
SELECT Borrower.Name AS Bname , LoanRequest.DateOfRequest AS DOR,
LoanRequest.Amount AS AM
FROM Borrower, LoanRequest
WHERE Borrower.BId=LoanRequest.BId
ORDER BY LoanRequest.Amount ASC
```

Query_4:

Συνολικό ποσό δέσμευσης:

```
SELECT Lender.Name AS Lname , SUM(Commitment.Amount) AS AM
FROM Lender,Commitment
WHERE Lender.LId=Commitment.LId
GROUP BY Lender.LId DESC
```

Query_5:

Lenders με το μεγαλύτερο ποσοστό συμμετοχής:

```
SELECT Lender.Name AS LN , Trust.Percentage AS TP
FROM Trust , Lender , Borrower
WHERE Borrower.BId=Trust.BId AND Trust.LId = Lender.LId
     AND Trust.Percentage= (SELECT MAX(Trust.Percentage) FROM Trust)
```

Query_6:

Ταξονόμηση μεσαζώντων-δανείων με βάση την ημερομηνία αίτησης:

```
SELECT Intermediary.MId AS M,Intermediary.Name AS N,Loan.Id AS ID
FROM Intermediary
INNER JOIN Loan
ON Intermediary.MId=Loan.MId
ORDER BY Loan.DateOfRequest
```

Query_7:

Lenders και συνολικό ποσό δέσμευσης αυτών αν το ποσό αυτό έιναι >=1000 :

```
SELECT Lender.Name AS L, SUM(Commitment.Amount) AS MYS
FROM Lender
INNER JOIN Commitment
ON Lender.LId=Commitment.Lid
GROUP BY Lender.Name
HAVING SUM(Commitment.Amount)>=15000;
```

Query_8:

Μετράει όλα τα δάνεια για κάθε Borrower και ταξινομεί με βάση το όνομα:

```
SELECT Borrower.Name AS Bname , COUNT(LoanRequest.BId) AS LR
FROM LoanRequest
LEFT JOIN Borrower
ON Borrower.BId=LoanRequest.BId
GROUP BY Borrower.Name;
```

Query_9:

Ταξινόμηση Borrower-Αίσηση-Deadline με βάση το Deadline:

```
SELECT Borrower.Name AS Bname , LoanRequest.DateOfRequest AS DOR
,LoanRequest.DeadLine AS DEAD
FROM Borrower, LoanRequest
WHERE Borrower.BId=LoanRequest.BId
ORDER BY LoanRequest.DeadLine ASC
```

Query_10:

Μέσος όρος trust του Lender προς τον Borrower:

```
SELECT Lender.Name AS Lname , AVG(Trust.Percentage) AS TP
FROM Borrower, Trust , Lender
WHERE Trust.LId = Lender.LId
GROUP BY Lender.Name
ORDER BY AVG(Trust.Percentage) DESC
```