
# Βάσεις Δεδομένων
# Αναφορά Εξαμηνιαίας Εργασίας

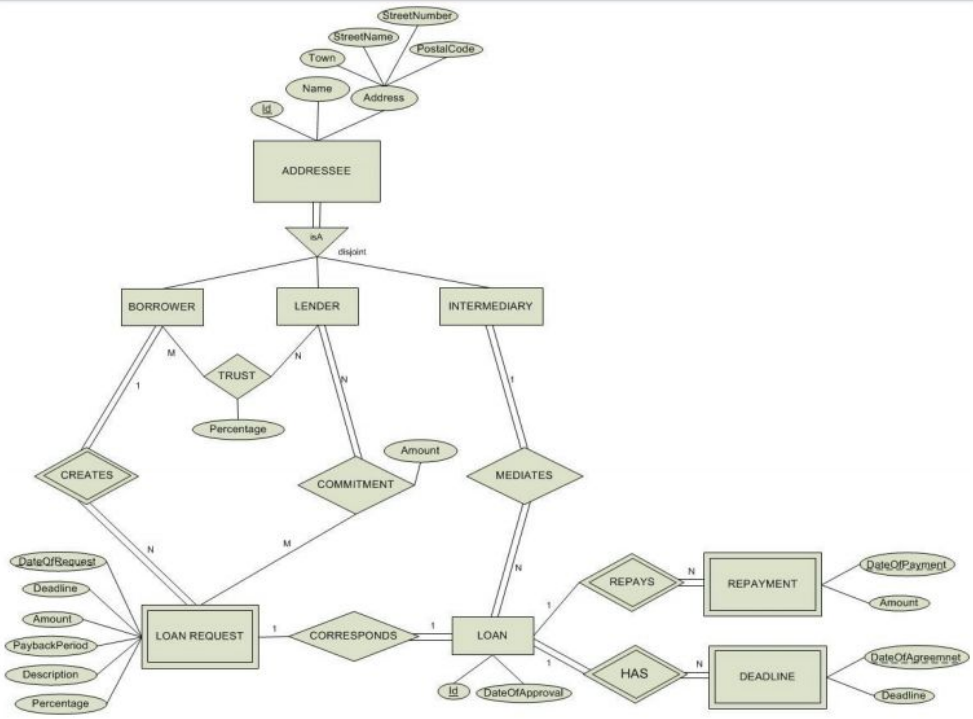
Χαρδούβελης Παναγιώτης-Ιάσων
03111082
Παππάς Σπυρίδων
03111154
Καψούλης Νικόλαος
03111110
9ο Εξάμηνο ΣΗΜΜΥ

# ΖΗΤΗΜΑΤΑ

a)
Στο πλαίσιο του παρόντος project υλοποιήθηκε η Βάση Δεδομένων Σχεσιακού μοντέλου που δίνεται ως λύση στη σελίδα του μαθήματος. Κατά το στάδιο της υλοποίησης παρατηρήθηκαν συγκεκριμένα πλεονεκτήματα και μειονεκτήματα της πλατφόρμας MySQL, την οποία χρησιμοποιήσαμε για την κατασκευή της βάσης.
Η MySQL αποτελεί μια απλή πλατφόρμα για το στήσιμο βάσεων δεδομένων. Η εγκατάσταση και η εφαρμογή των εντολών κονσόλας γίνεται εξαιρετικά απλή για τον χρήστη διαδικασία, ενώ είναι και lightweight πρόγραμμα. Επίσης, υπάρχει πολύ documentation ελεύθερα αναγνώσιμο το οποίο βοηθάει οποιονδήποτε δεν είχε ποτέ επαφή να καταλάβει βασικά πράγματα και να προχωρήσει σταδιακά στο στήσιμο μιας αληθινής βάσης δεδομένων. Ακόμη, είναι open-source και συμβατή με όλα τα βασικά web εργαλεία όπως html, css, php και html5, με την έννοια ότι μπορούμε να γράφουμε sql queries μέσα σε παραπάνω κώδικα, καλώντας πρώτα τη βάση δεδομένων με console εντολή. Για την οργάνωση και τη σύνδεση όλων των παραπάνω σε GUI χρησιμοποιήσαμε Bootstrap. Όλα τα παραπάνω μάς βοήθησαν στην υλοποίηση της βάσης και αποτελούν έναν από τους πιο συνηθισμένους συνδιασμούς εργαλείων για το στήσιμο SQL Websites.
Ένα άλλο πολύ σημαντικό πλεονέκτημα είναι ότι το mysql κατασκευάζει τα indexes αυτόματα για μια καινούργια σχέση, ένω γίνεται και ρητά με εντολή.

Από την άλλη, η MySQL πλατφόρμα περιορίζει σε ορισμένα σημεία. Για παράδειγμα, η οργάνωσή της σε tables με υποχρεωτικές τιμές πεδίων, οδηγεί σε μεγάλη σπατάλη μνήμης όταν χρησιμοποιούνται πίνακες με πολλές NULL τιμές (λύση προβλήματος με NoSQL databases - για τεράστια συστήματα), ενώ επίσης η συγκεκριμένη πλατφόρμα είναι αποδεδειγμένο ότι δεν λειτουργεί κανονικά σε μεγάλα συστήματα, παρόλο που θεωρείται scalable μέχρι και τα 8TB μνήμης. Ακόμα, δεν γινόταν να δημιουργήσουμε ενα table στην περίπτωση που ένα foreign key του αναφερόνταν σε collumn που δεν είναι πρώτο στην αντίστοιχη σχέση. Αυτό το πρόβλημα αντιμετωπίστηκε με τη χρήση της μηχανής INNODB (ENGINE=INNODB) και την δήλωση ενός επιπλέον index στο επίμαχο πεδίο (CREATE INDEX date_index on LoanRequest (DateOfRequest) USING BTREE;).
Τέλος, σε αυτό το σημείο θα θέλαμε να προσθέσουμε ότι παρόλο που η SQL είναι τυπικά open source, τα τελευταία χρόνια ελέγχεται εξ ολοκλήρου από την Oracle - αλλά αυτό αφορά περισσότερο τον developer και όχι έναν απλό user.

b)
Το Σχεσιακό Μοντέλο με βάση το οποίο υλοποιήσαμε τη βάση είναι το προτεινόμενο (λύση).




Ακολουθούν τα constraints που έχουμε ορίσει:

Οποιαδήποτε διαγραφή Borrower αυτομάτως διαγράφει τα αντίστοιχα Loan Requests, Trust και Commitment entries γι αυτόν τον Borrower αλλά όχι και το αντίστοιχο loan γιατί επιλεγουμε να μενει στη βάση.

Οποιαδήποτε διαγραφή Lender αυτομάτως διαγράφει τα αντίστοιχα Commitment και Trust.

Οποιαδήποτε διαγραφή Loan αυτομάτως διαγράφει τα αντίστοιχα Deadlines.

Κατά την εισαγωγή καινούργιου borrower, lender h intermediary, δεν ορίσαμε τα κλειδιά ως auto-increment αλλά με ένα παραπάνω query στη βάση βρίσκαμε το max id και το καινούργιο entry έπερνε ως id :=max_id+1.

Το mysql δημιουργεί αυτόματα τα indexes βάσει των primary keys που δηλώνονται σε κάθε σχέση ενώ στην σχέση LoanRequest έχουμε ζητήσει και εμείς ρητα τη δημιουργία ενός index για το γνώρισμα DateOfRequest καθώς το συγκεκριμένο γνώρισμα υπάρχει ως foreign key σε πολλές άλλες σχέσεις.

c)
Το αρχείο που κατασκευάζει τη βάση καθώς και μερικές τιμές για κάθε σχέση υπάρχουν σε αρχεία που επισηνάπτουμε για ευκολία μαζί με την αναφορά. Συγκεκριμένα η δημιουργία της βάσης γίνεται με το αρχείο microloans.sql, ενώ τα δύο triggers και τα δύο views βρίσκονται στο τέλος της αναφοράς.

## QUERIES

Τα περισσότερα από τα παρακάτω queries δέχονται Inputs από τον χρήστη μέσω του GUI και με χρήση της php . Τα queries που παρουσιάζουμε εδώ αντί για μεταβλητή παίρνουν μια τυχαία τιμή ως όρισμα για να είναι ευανάγνωστα.

---

**Επιστρέφει Id, όνομα και trust (επι τοις εκατό) που έχει ένας Lender προς τους Borrower με τους οποίους συνεργάζεται.**

**input : LId**

```
select  Borrower.BId, Borrower.Name, Percentage
from Borrower, Trust
where 2=Trust.LId and Borrower.BId=Trust.BId
order by Percentage DESC
```

---

**Επιστρέφει Id, Ημερομηνία αποπληρωμής και αποπληρωτέο ποσό, για όλα τα Repayments ενός συγκεκριμένου Borrower, δηλαδή το ιστορικό των πληρωμών ενός Borrower.**
**input :BId**

```
select * from Repayment
where Id in (select Id from Loan where BId = 13)
order by Id
```

---

**Επιστρέφει το μέγιστο και το ελάχιστο ποσό προς αποπληρωμή από την λίστα δανείων (συμπεριλαμβανομένου του επιτοκίου) καθώς και τον μέσο όρο των ποσών για αποπληρωμή.**

```
select max(X), min(X), avg(X)
from
(select Loan.Id, (LoanRequest.Amount+ LoanRequest.Amount*Percentage) as X
from Loan,LoanRequest
```

```
where Loan.BId = LoanRequest.BId and LoanRequest.DateOfRequest = Loan.DateOfRequest
order by Loan.Id
)as A
```

**/*to posa lefta xrostaei kathe borrower (sum)*/**

```
select A.BId, X-Y
from
(
select Loan.BId, sum(LoanRequest.Amount+ LoanRequest.Amount*Percentage)
as X
from Loan,LoanRequest
where Loan.BId = LoanRequest.BId and LoanRequest.DateOfRequest =
Loan.DateOfRequest
group by Loan.BId
) as A,
(
select BId, sum(Amount) as Y
from   Repayment, Loan
where  Loan.Id = Repayment.Id
group by BId
) as B
where A.BId = B.BId

union /*autoi pou dn exoun plirosei  tipota*/

select Loan.BId, sum(LoanRequest.Amount+ LoanRequest.Amount*Percentage)
from Loan,LoanRequest, Repayment
where Loan.BId = LoanRequest.BId and LoanRequest.DateOfRequest =
Loan.DateOfRequest
and Loan.Id not in (select distinct Id from Repayment)
group by Loan.BId
```

---

**Επιστρέφει το deadline των δανείων ενός συγκεκριμένου Borrower μέχρι κάποια ημερομηνία και τα διατάσσει με βάσει το πιο πρόσφατο DateOfAgreement.**
**input : BId**

```
select A.Id, DateOfAgreement, deadline from
(
select Id, DateOfAgreement, max(Deadline) as deadline
from Deadline
group by Id
having deadline < '2021-01-01'
```

```
) as A, Loan
where A.Id = Loan.Id and Loan.BId = 13 /*apo input*/
```

---

**Για κάποιον Borrower το max trust του καθώς και το πόσα δάνεια έχει πάρει.**
**input: BId**

```
select 13 BId, (select Name from Borrower where BId=13) Name,
avg(Trust.Percentage) AverageTrust,
(select count(Id)
from Loan
where BId = 13)  NumofLoans
from Trust
group by BId
having BId = 13
```

---

**Επιστρέφει το ποσό που χρειάζεται για να γίνει μια αίτηση Loan καθώς και το αντίστοιχο deadline.**
**inputs : BId, DateOfRequest**

```
select LoanRequest.Deadline, (LoanRequest.Amount -
sum(Commitment.Amount)) leftover

from Commitment, LoanRequest

where 3 = Commitment.BId and '2015-01-01' = Commitment.DateOfRequest
and LoanRequest.DateOfRequest = '2015-01-01' and LoanRequest.BId=3
```

---

**Επιστρέφει τη λίστα με τις deadline-ημερομηνίες πληρωμής πριν απο κάποια ημερομηνία για κάποιον Lender.**
**inputs : Lender Id, date**

```
select A.Id, BId, deadline
from
(
select Id, DateOfAgreement, max(Deadline) as deadline
from Deadline
group by Id
having deadline < '2021-01-01' /*<----change this */
) as A, Loan

where A.Id in (
```

```
select Id from Loan,Commitment
where Commitment.LId = 18 and Loan.BId = Commitment.BId and
Loan.DateOfRequest = Commitment.DateOfRequest
) and Loan.Id = A.Id
```

---

**Όλα τα request που λήγουν σε μια ημερομηνία και θέλουν λιγότερο απο ένα συγκεκριμένο ποσό για να ικανοποιηθούν**

**inputs : Deadline-date,leftover**

```
select * from
(
select A.BId, A.DateOfRequest, B.Deadline, (Y - X) as leftover
from
(
select BId, DateOfRequest, sum(Amount) X
from Commitment
group by BId, DateOfRequest
) A,

(select BId, DateOfRequest, Deadline, Amount Y from LoanRequest order by
BId, DateOfRequest) B

where A.BId = B.BId and A.DateOfRequest = B.DateOfRequest
order by A.BId
) Skata
where Deadline<= '2015-01-01' and leftover < 100 and leftover>0
```

---

**Επιστρέφει το Id, την πόλη, το συνολικό ποσό και την περιγραφή που έχουν μέγιστο ποσό μεγαλύτερο από 150000.**

```
select Borrower.BId, Town, max(Amount) max, Description
from Borrower, LoanRequest
where Borrower.BId = LoanRequest.BId
group by Borrower.BId
having max>150000
```

---

## VIEWS

**Views:**

**/*not editable*/**

```
create view IntermediaryOverview as
select  Intermediary.MId, Name, count(Intermediary.MId) numofloans ,
avg(Amount) average
from Intermediary, Loan, LoanRequest
where Loan.MId = Intermediary.MId and Loan.DateOfRequest =
LoanRequest.DateOfRequest
and Loan.BId=LoanRequest.BId
group by Intermediary.MId
```

**/*editable*/**

```
create view TrustView as

select Borrower.BId, Borrower.Name, Percentage
from Borrower, Trust
where Borrower.BId=Trust.BId
order by Percentage DESC
```

## TRIGGERS

Σε περίπτωση που ένα δάνειο αποπληρωθεί πλήρως τότε διαγράφεται το αντιστοιχο entry στον πίνακα των loans. Επιλέγουμε να μη διαγραφούν τα entries sta deadlines k sta repaiments ώστε η “τράπεζα” να κρατάει αρχείο.

```
delimiter //
CREATE TRIGGER fullpay
AFTER INSERT ON Repayment
FOR EACH ROW
BEGIN
IF (
    SELECT sum(Amount)
    FROM Repayment
    WHERE New.Id=Repayment.Id)
     >=
    (SELECT (Amount+Percentage*Amount)
    FROM LoanRequest AS LR,Loan AS L
    WHERE LR.DateOfRequest=L.DateOfRequest and LR.BId=L.BId and
L.Id=New.Id
) THEN
```

```
DELETE FROM Loan WHERE Loan.Id=New.Id;
END IF;
END; //
delimiter ;
```

Το παρακάτω trigger τσεκάρει ότι δεν δημιουργούνται ασυνέπειες στο σύστημα σε περίπτωση που ο χρήστης εισάγει αρνητικό ποσό πληρωμής.

```
CREATE TRIGGER ins_check BEFORE INSERT ON Repayment
FOR EACH ROW
BEGIN
IF NEW.Amount < 0 THEN
SET NEW.Amount = 0;
END IF;
END;
```