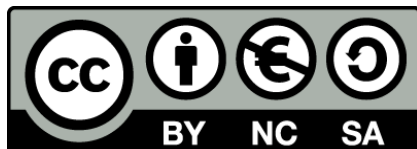




## διδασκτικές ενότητες στις Βάσεις Δεδομένων

Επανάληψη και λυμένα θέματα του μαθήματος «Βάσεις Δεδομένων ΙΙ» για τη διδασκαλία του στο Τμήμα Πληροφορικής του ΤΕΙ Αθήνας.

Διδάσκων: **Χ. Σκουρλάς**, [cskourlas@teiath.gr](mailto:cskourlas@teiath.gr)



## Βάσεις Δεδομένων II – test Exam#1



Εργάζεστε για την εταιρεία Mythical Films, μια εταιρεία ενοικίασης ταινιών, και σας ανατίθεται ο σχεδιασμός της Βάσης Δεδομένων των ταινιών της εταιρείας.

### Περιορισμοί

1. Η εταιρία Mythical films διαθέτει ταινίες σε διάφορες κατηγορίες (FilmCategories): Comedy, Drama,....
2. Κάθε ταινία ανήκει σε μια κατηγορία.
3. Για κάθε ταινία θα υπάρχουν απαραίτητα υπότιτλοι στα αγγλικά. Επιπλέον οι υπότιτλοι μπορούν να υπάρχουν είτε στα ελληνικά είτε στα γαλλικά είτε και στα δύο.
4. Για κάθε ταινία έχουμε καταχωρημένους και τους ηθοποιούς.

### Ασκήσεις

- Για την παραπάνω περιγραφή δημιουργείστε τους κατάλληλους πίνακες .
- Γράψτε όλες τις παραδοχές περιορισμούς που θέσατε (π.χ. ένας ηθοποιός μπορεί να παίζει σε πολλές ταινίες).
- Δημιουργείστε τον κατάλληλο trigger ο οποίος θα ενημερώνει τον αριθμό των ταινιών στις κατηγορίες (no\_of\_films) μετά απο κάθε ενημέρωση του πίνακα των ταινιών.
- Δημιουργήστε μια procedure η οποία θα δέχεται σαν είσοδο το όνομα της κατηγορίας και θα επιστρέφει το σύνολο των ταινιών, και τις γλώσσες που υποστηρίζουν οι ταινίες της συγκεκριμένης κατηγορίας.

Κατηγορία: Drama

Σύνολο Ταινιών	Υπότιτλοι στα Αγγλικά	Υπότιτλοι στα Ελληνικά	Υπότιτλοι στα Γαλλικά
20	20	8	1



Εργάζεστε για το ταξιδιωτικό γραφείο Blue Moon, και σας ανατίθεται ο σχεδιασμός της Βάσης Δεδομένων των τουριστικών πακέτων της εταιρείας.

### Περιορισμοί

1. Η εταιρία διαθέτει πολλές κατηγορίες προορισμών: Romantic, Winter ,....
2. Κάθε τουριστικό πακέτο μπορεί να ανήκει σε παραπάνω από μια κατηγορίες.
3. Για κάθε πακέτο θα έχουμε καταχωρημένο τον τρόπο μετάβασης στον προορισμό (αεροπλάνο, πλοίο, λεωφορείο) και τη διάρκεια σε ημέρες.
4. Για κάθε πακέτο θα υπάρχει και το αντίστοιχο κόστος σε ευρώ.

### Ασκήσεις

- Για την παραπάνω περιγραφή δημιουργήστε τους κατάλληλους πίνακες .
- Γράψτε όλες τις παραδοχές-περιορισμούς που θέσατε π.χ. σε έναν προορισμό μπορείτε να πάτε με περισσότερα απο ένα μέσα μεταφοράς . λεωφορείο ή αεροπλάνο.
- Δημιουργήστε μια procedure η οποία θα δέχεται σαν είσοδο την κατηγορία του τουριστικού πακέτου, τον αριθμό των ημερών και το μεταφορικό μέσο και θα επιστρέφει τα πακέτα που υπάρχουν διαθέσιμα.
- Δημιουργήστε μια function η οποία θα μετατρέπει το κόστος του πακέτο απο ευρώ σε δολάρια. Θεωρείστε οτι 1 ευρώ=1.20 δολάρια.

## Βάσεις Δεδομένων II – test Exam#3



Εργάζεστε για την ασφαλιστική εταιρεία **General Insurance**, και σας ανατίθεται ο σχεδιασμός και η υλοποίηση της Βάσης Δεδομένων των ασφαλιστικών προϊόντων - πακέτων της εταιρείας.

### Περιορισμοί

1. Η εταιρεία διαθέτει πολλά ασφαλιστικά προϊόντα (InsuranceCovers): Health Insurance, Critical Illness Cover, Home insurance, Car insurance, ...
2. Κάθε ασφαλισμένος (customer) μπορεί να αγοράζει παραπάνω από ένα ασφαλιστικά προϊόντα, δηλαδή να υπογράφει πολλά συμβόλαια (contracts) ένα για κάθε ασφαλιστικό προϊόν που αγοράζει.
3. Για κάθε προϊόν θα έχουμε καταχωρημένα την ημερομηνία έναρξης (start) και λήξης (end) του καθώς επίσης και το κόστος (amount) του.

### Ασκήσεις

- Για την παραπάνω περιγραφή δημιουργήστε τους κατάλληλους πίνακες .
- Γράψτε όλες τις παραδοχές- περιορισμούς που θέσατε.
- Δημιουργήστε triggers που θα ενημερώνουν αυτόματα τον αριθμό των συμβολαίων ανά πελάτη (no\_of\_contracts)
- Δημιουργήστε μια procedure η οποία θα δέχεται σαν είσοδο το όνομα ενός πελάτη και θα επιστρέφει τον αριθμό των συμβολαίων που κατέχει καθώς επίσης και το συνολικό κόστος.
- Δημιουργήστε μια function η οποία θα επιστρέφει τη διάρκεια των συμβολαίων λαμβάνοντας υπόψιν την ημερομηνία έναρξης και λήξης του συμβολαίου.(π.χ. 6 μήνες).

## Βάσεις Δεδομένων II – test Exam#4



### Εργασία

Εργάζεστε για την αεροπορική εταιρία Chartered Airplane, και σας ανατίθεται ο σχεδιασμός της Βάσης Δεδομένων των κρατήσεων της εταιρείας.

### Περιορισμοί

1. Η εταιρία πραγματοποιεί πτήσεις εσωτερικού και εξωτερικού.
2. Για κάθε πτήση (flight) καταχωρούνται ο κωδικός της (flight\_no), το αεροδρόμιο αναχώρησης (departure) και το αεροδρόμιο άφιξης (arrival) και ο αριθμός διαθέσιμων θέσεων (seats).
3. Η εταιρία καταχωρεί τις κρατήσεις (reservations) των πελατών (customers): αρ. κράτησης, στοιχεία πελάτη,...

### Ασκήσεις

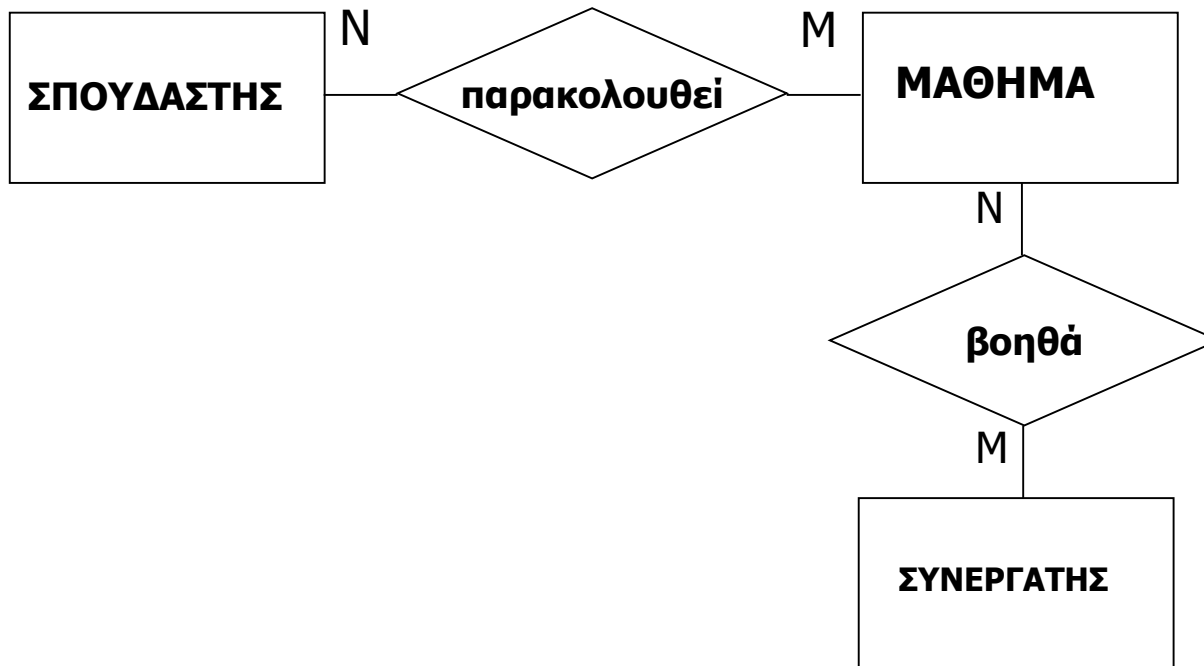
- Για την παραπάνω περιγραφή δημιουργήστε τους κατάλληλους πίνακες .
- Γράψτε όλες τις παραδοχές-περιορισμούς που θέσατε.
- Δημιουργήστε μια procedure η οποία θα δέχεται σαν είσοδο το όνομα ενός αεροδρομίου και θα επιστρέφει τον αριθμό των κρατήσεων.
- Δημιουργήστε ένα trigger, ο οποίος για κάθε κράτηση που θα πραγματοποιείται θα μειώνει κατά ένα τον αριθμό των διαθέσιμων θέσεων.

# Μοντελοποίηση

# Βαθμός Συσχέτισης

- ◆ Βαθμός μιας συσχέτισης ονομάζεται ο αριθμός των οντοτήτων που συνδέει.
- ◆ Συνήθως οι συσχετίσεις μεταξύ δύο οντοτήτων (δυαδικές συσχετίσεις) επαρκούν για τις ανάγκες μεγάλου μέρους της εφαρμογής.
- ◆ Υπάρχουν περιπτώσεις όπου τρεις ή περισσότερες οντότητες πρέπει να συνδεθούν με μια συσχέτιση ή μια συσχέτιση να οριστεί πάνω σε οντότητα(ες) και συσχέτιση(εις).

## Βαθμός Συσχέτισης

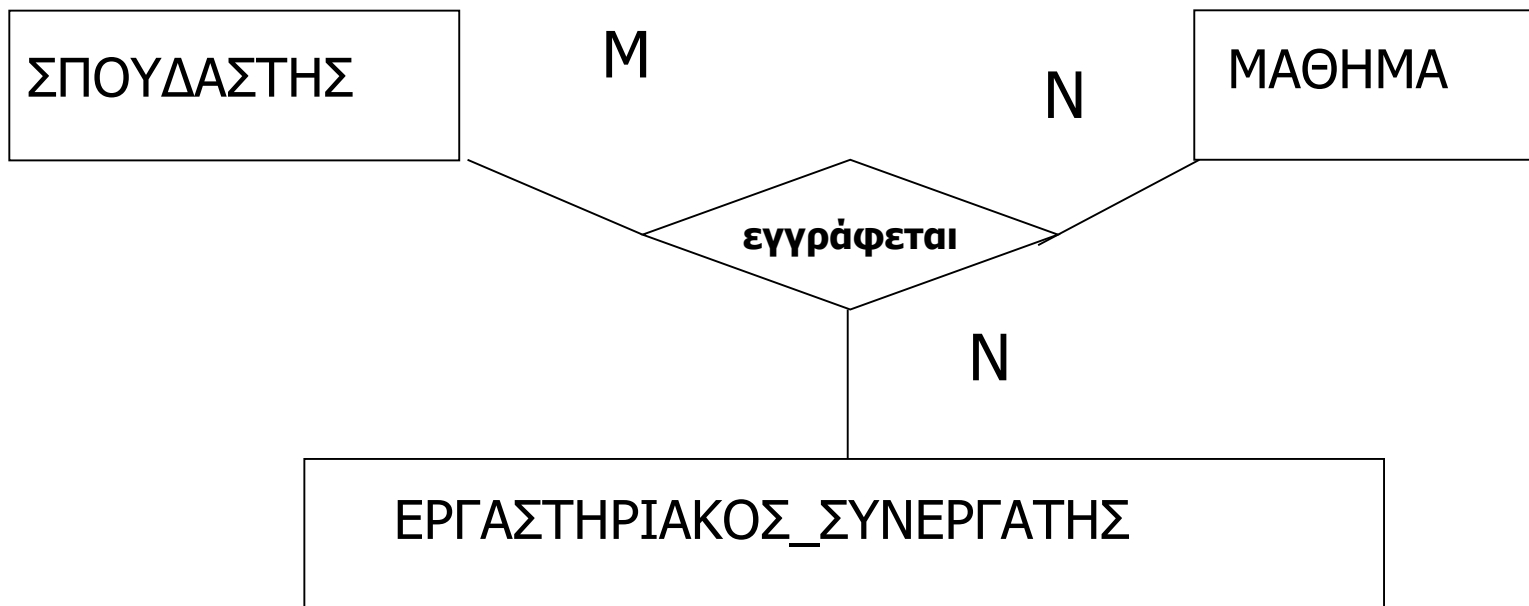


Διαδικές  
Συσχετίσεις

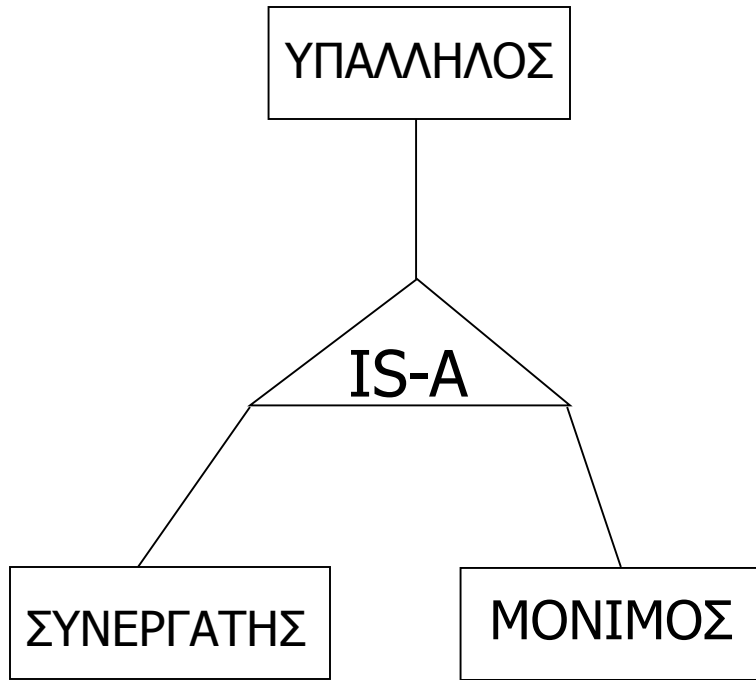
- Το μοντέλο είναι επαρκές; Ναι αν όλοι οι εργαστηριακοί συνεργάτες βοηθούν όλους τους σπουδαστές.
- Τι γίνεται, όμως, αν οι σπουδαστές ανήκουν σε εργαστηριακά τμήματα και σε κάθε τμήμα είναι υπεύθυνος ένας και μόνο εργαστηριακός συνεργάτης;



# Τριαδική Συσχέτιση



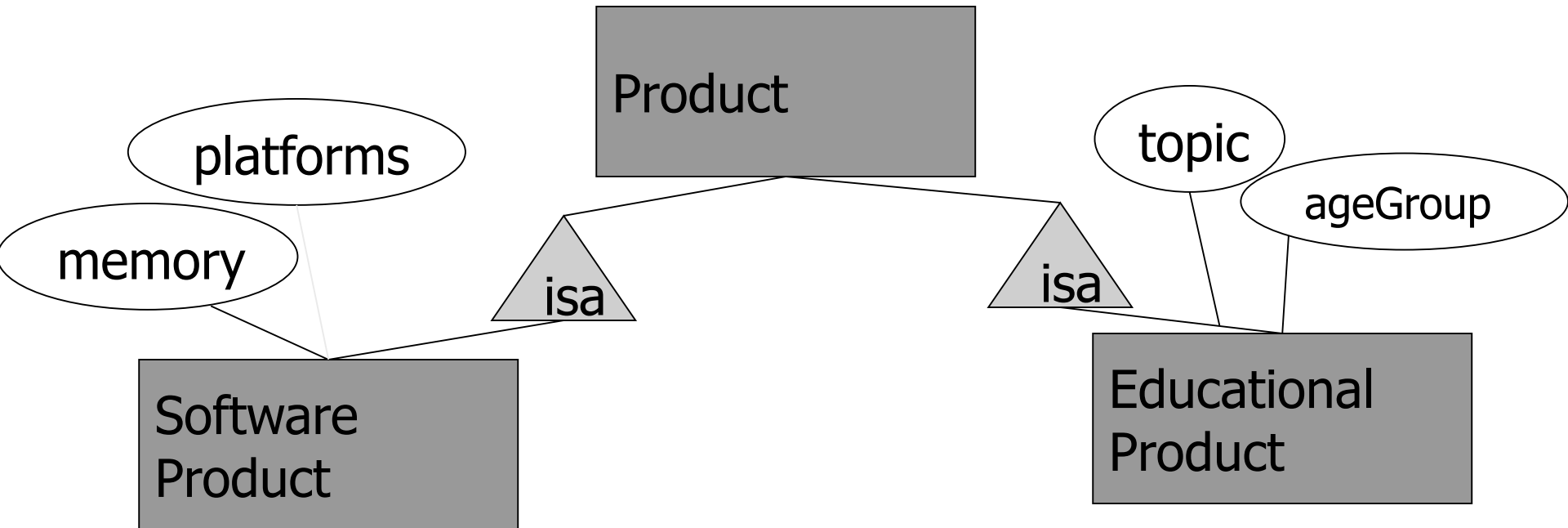
## Συσχέτιση «Is-A»



**Disjoint and Complete mapping**

- Κάθε «ΕΚΤΑΚΤΟΣ» και κάθε «ΜΟΝΙΜΟΣ» θεωρείται και «ΥΠΑΛΛΗΛΟΣ» δηλαδή κληρονομεί όλα τα χαρακτηριστικά της οντότητας «ΥΠΑΛΛΗΛΟΣ»
- Χρειάζεται πολλές φορές να εκφράσουμε μια οντότητα ως «εξειδίκευση» (specialization) μιας άλλης

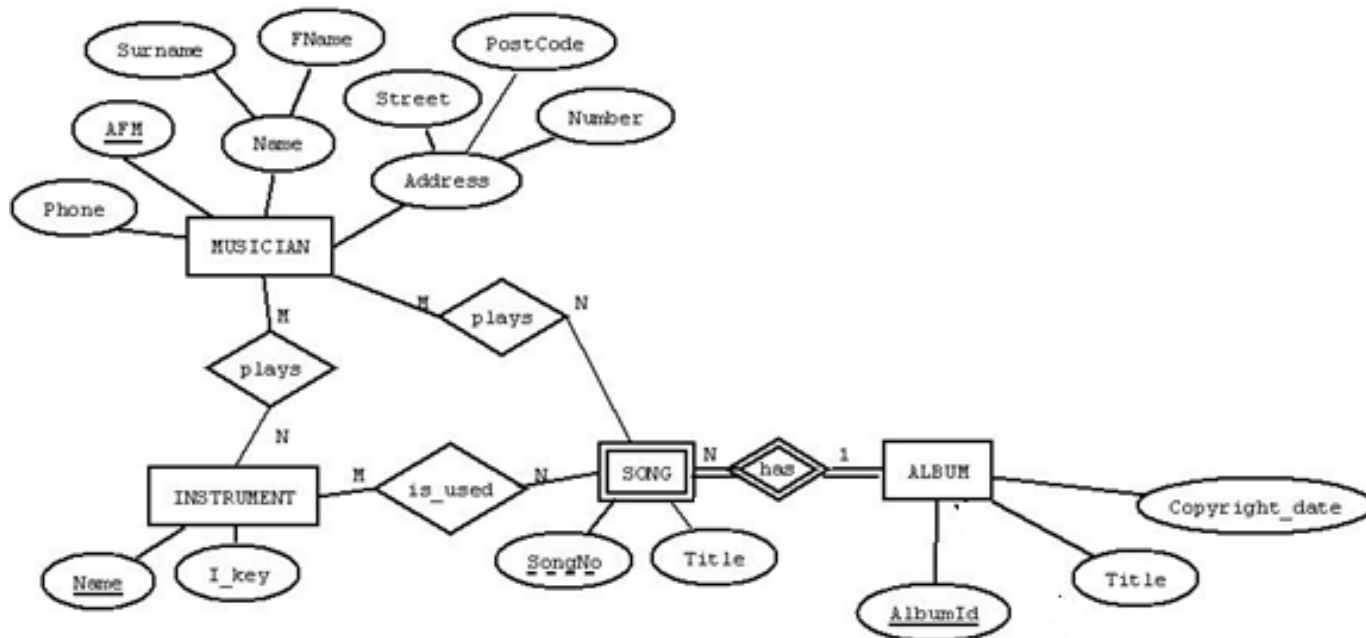
# Πώς να μεταγράψουμε υποκλάση (How to translate a subclass)



## Επιλογή 1 (Option 1): The E/R Approach

- ◆ Product (name, price, category, manufacturer)
- ◆ EducationalProduct (name, **ageGroup**, **topic**)
- ◆ SoftwareProduct (name, **platforms**, **requiredMemory**)
- ◆ Θυμηθείτε ότι το ίδιο όνομα στήλης μπορεί να εμφανίζεται σε πολλές σχέσεις (Same name may appear in several relations) παρά το γεγονός ότι κάθε φορά εκφράζει ενδεχομένως κάτι διαφορετικό

Κάθε μουσικό όργανο χρησιμοποιείται για την εγγραφή ενός ή περισσότερων τραγουδιών και για την εγγραφή ενός τραγουδιού χρησιμοποιούνται ένα ή περισσότερα όργανα. Στο μοντέλο θεωρούμε ακόμη ότι ένας μουσικός μπορεί να παίζει σε πολλά τραγούδια και σε ένα τραγούδι παίζουν πολλοί μουσικοί. **Ανασχεδιάστε στην περίπτωση που μας ενδιαφέρει να γνωρίζουμε για κάθε τραγούδι ποια όργανα έπαιξε κάθε μουσικός. Ποια τα ξένα κλειδιά στους αντίστοιχους πίνακες;**



Στο τμήμα μας χρησιμοποιούνται 2 συστήματα βάσεων δεδομένων, το μαθητολόγιο και το μητρώο καθηγητών. Κατασκευάστε την κανονική μορφή **Boyce-Codd** για ενιαίο σύστημα βάσης δεδομένων που θα αντικαταστήσει τα δύο συστήματα.

Μαθητολόγιο

student

Studno	Sname	Address	Semester
10	CODD	ATHENS	C
20	MARTIN	ATHENS	D
30	DATE	BERLIN	C

Student\_lesson

Studno	Lesson	mark
10	100	75
10	200	25
20	100	30
20	200	70
30	100	100

Lesson

L_code	Lesson
100	DATABASE I
200	DATABASE II

## Μητρώο καθηγητών

### Teacher

Tno	tname	Address	Speciality
100	CODD	ATHENS	DATABASE
200	MARTIN	ATHENS	BUSSINESS INTELLIGENCE
300	DATE	BERLIN	DATABASE

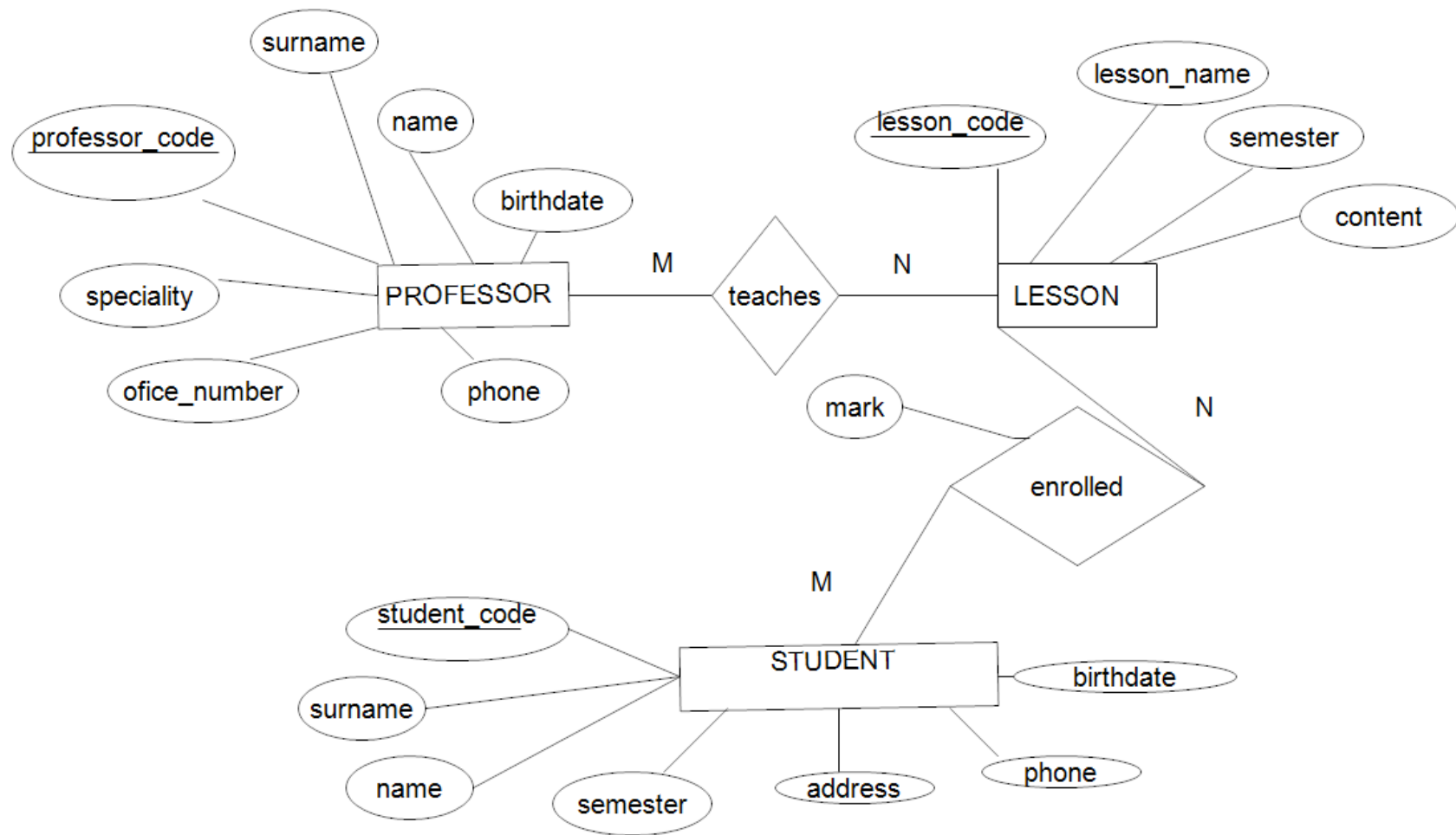
### Teaches

Studno	Lesson
100	100
100	200
200	100
200	200
300	100

### Lesson

L_code	Lesson	Semester
100	DATABASE I	C
200	DATABASE II	D

Μοντελοποιήστε τη βάση δεδομένων του ενιαίου συστήματος.





Έστω απλοποιημένη βάση δεδομένων γονέων. Οι στήλες του πίνακα αυτού είναι οι εξής:

Empno=Κωδικός υπαλλήλου, Name=όνομα, JobNo=κωδικός θέσης, Job=θέση, Deptno=κωδικός τμήματος, Dname=τμήμα Sal=μισθός, C\_No=αριθμός παιδιών υπαλλήλου, C\_Name=όνομα παιδιού, B\_Date= ημερομηνία γέννησης παιδιού.

## Περιορισμοί

Υποτίθεται ότι κάθε υπάλληλος έχει μία θέση, ανήκει σε ένα τμήμα, ο μισθός του εξαρτάται από τη θέση και έχει παιδιά (λίγο περιοριστικό αυτό).

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ	10-JAN-89
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΙΩΑΝΝΗΣ	20-MAR-90
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10-JUN-89

Κύριο κλειδί: (empno, c\_name) Θυμηθείτε τους κανόνες ακεραιότητας

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΜΑΡΙΑ	10-JAN-89
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	ΙΩΑΝΝΗΣ	20-MAR-90
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000			
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	ΘΩΜΑΣ	10-JUN-89

Κύριο κλειδί: ????? Accno

## Δεύτερη Κανονική Μορφή 2NF

*empno* -- >

*c\_name*) -- >

(*empno*, *c\_name*) -- >

### Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000	
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1

Κύριο κλειδί: (*empno*)

### Child

Empno	C_Name	B_date
10	ΜΑΡΙΑ	10-JAN-89
10	ΙΩΑΝΝΗΣ	20-MAR-90
30	ΘΩΜΑΣ	10-JUN-89

Κύριο κλειδί: (*empno*, *c\_name*)

### Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

Κύριο κλειδί: (*c\_name*)

## Τρίτη Κανονική Μορφή 3NF

### Employee

Empno	Name	JobNo	DeptNo	C_no
10	ΣΠΥΡΟΥ	100	50	2
20	ΧΡΗΣΤΟΥ	200	60	
30	ΝΙΚΟΥ	300	70	1

*Κύριο κλειδί: empno*

### Jobs

JobNo	Job	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

*Κύριο κλειδί: JobNo*

### Child

Empno	C_Name	B_date
10	ΜΑΡΙΑ	10-JAN-89
10	ΙΩΑΝΝΗΣ	20-MAR-90
30	ΘΩΜΑΣ	10-JUN-89

*Κύριο κλειδί: (empno, c\_name)*

### Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

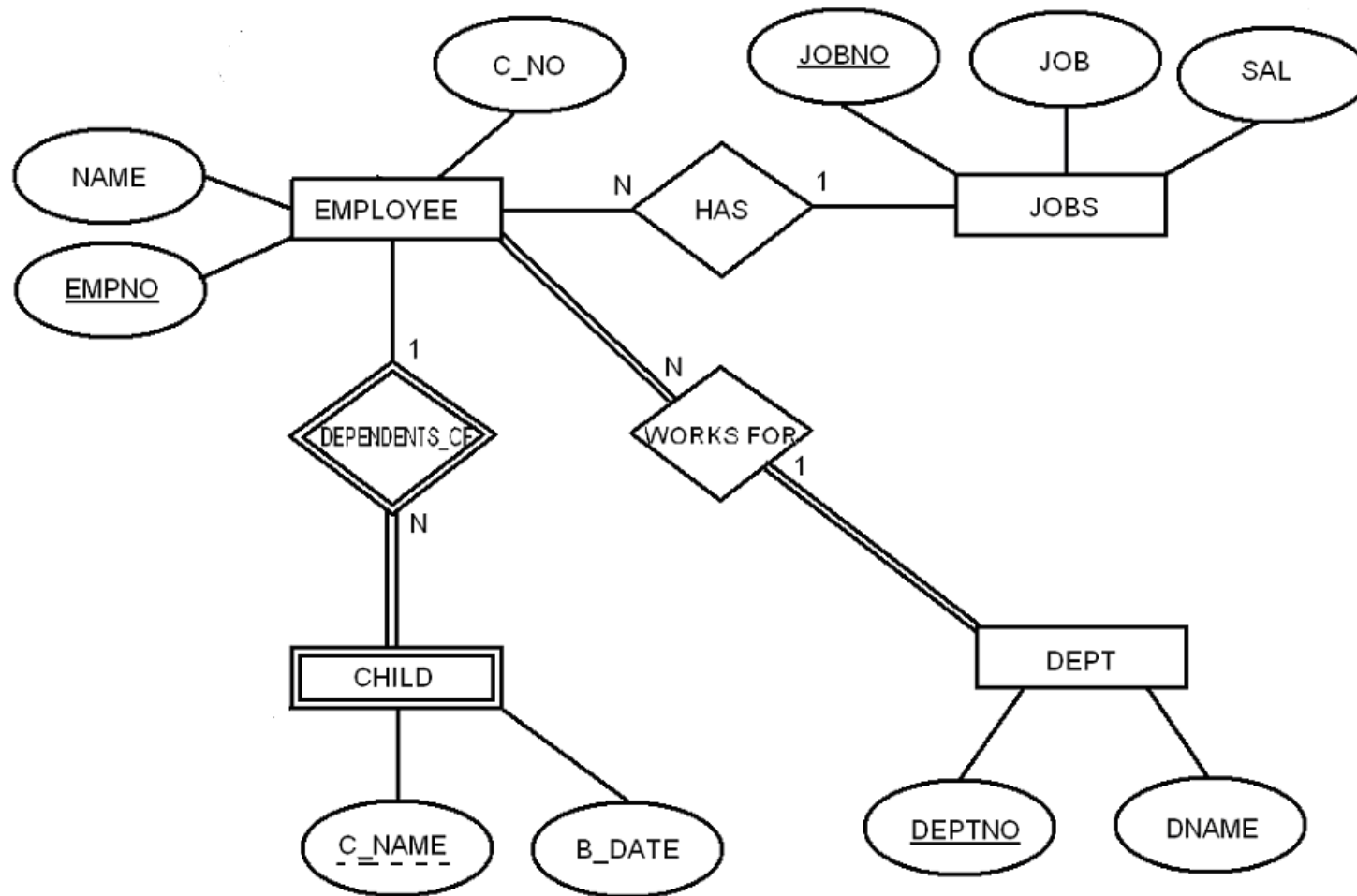
*Κύριο κλειδί: (c\_name)*

### Dept

DeptNo	Dname
50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

*Κύριο κλειδί: deptno*

Μοντέλο Οντοτήτων Συσχετίσεων (Entity Relationship model).



Η στήλη Ch\_No συμβολίζει το μοναδικό αριθμό κάθε παιδιού.

Employee

Empno	Name	JobNo	Job	DeptNo	Dname	Sal	C_no	Ch_no	C_Name	B_date
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	1	ΜΑΡΙΑ	10-JAN-89
10	ΣΠΥΡΟΥ	100	ΠΩΛΗΤΗΣ	50	ΠΩΛΗΣΕΙΣ	2200	2	2	ΙΩΑΝΝΗΣ	20-MAR-90
20	ΧΡΗΣΤΟΥ	200	ΑΝΑΛΥΤΗΣ	60	ΛΟΓΙΣΤΗΡΙΟ	2000				
30	ΝΙΚΟΥ	300	ΧΕΙΡΙΣΤΗΣ	70	ΜΙΣΘΟΔΟΣΙΑ	1000	1	3	ΘΩΜΑΣ	10-JUN-89

*Είναι το (ch\_no) Κύριο κλειδί?*

Η πρώτη και η δεύτερη κανονική μορφή συμπίπτουν?

## Τρίτη Κανονική Μορφή 3NF

### Employee

Empno	Name	JobNo	DeptNo	C_no
10	ΣΠΥΡΟΥ	100	50	2
20	ΧΡΗΣΤΟΥ	200	60	
30	ΝΙΚΟΥ	300	70	1

*Κύριο κλειδί: empno*

### Jobs

JobNo	Job	Sal
100	ΠΩΛΗΤΗΣ	2200
200	ΑΝΑΛΥΤΗΣ	2000
300	ΧΕΙΡΙΣΤΗΣ	1000

*Κύριο κλειδί: JobNo*

### Child

Empno	Ch_no	C_Name	B_date
10	1	ΜΑΡΙΑ	10-JAN-89
10	2	ΙΩΑΝΝΗΣ	20-MAR-90
30	3	ΘΩΜΑΣ	10-JUN-89

*Κύριο κλειδί: (ch\_no), foreign key: (empno)*

### Names

C_Name
ΘΩΜΑΣ
ΙΩΑΝΝΗΣ
ΜΑΡΙΑ

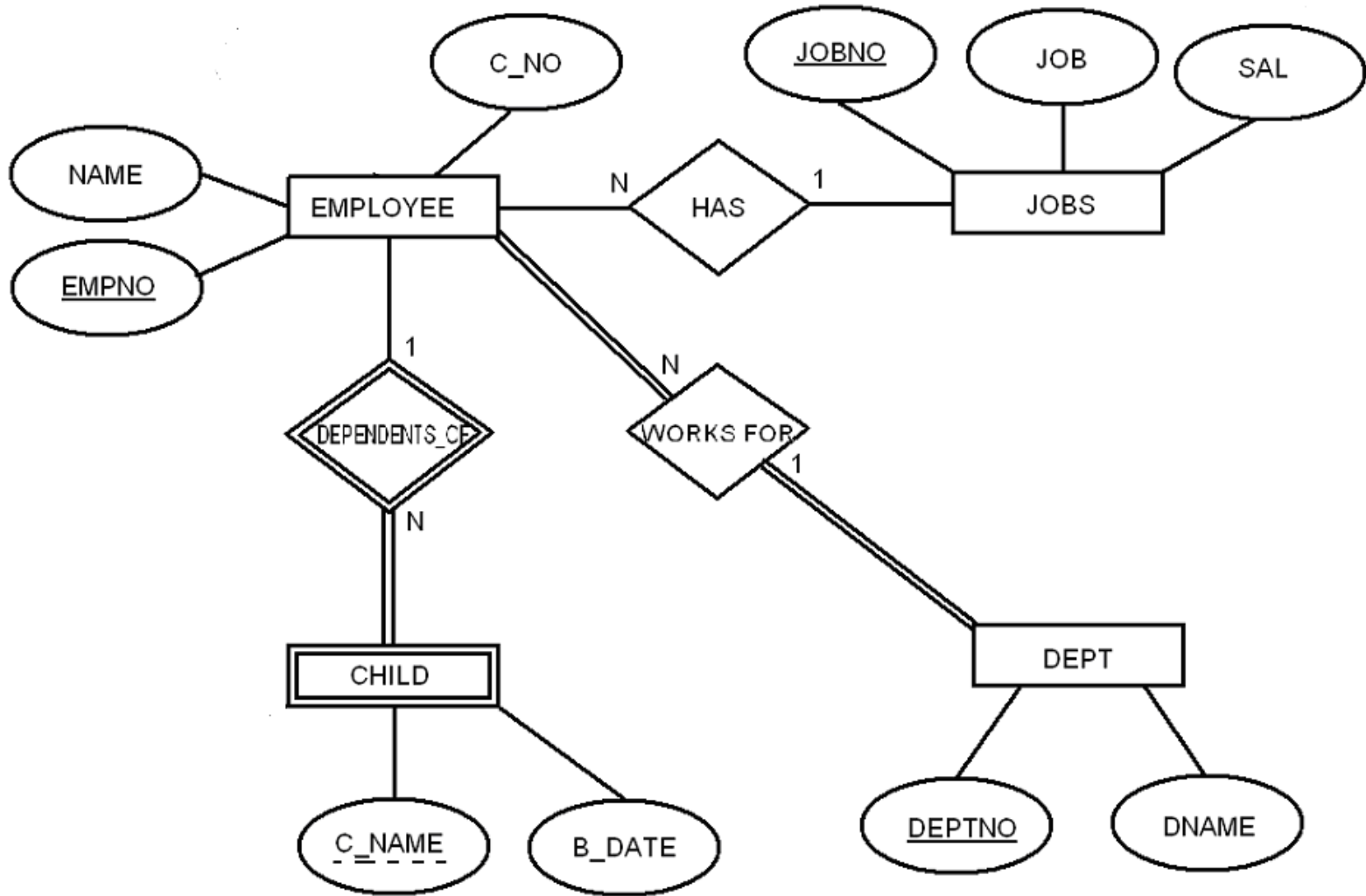
*Κύριο κλειδί: (c\_name)*

### Dept

DeptNo	Dname
50	ΠΩΛΗΣΕΙΣ
60	ΛΟΓΙΣΤΗΡΙΟ
70	ΜΙΣΘΟΔΟΣΙΑ

*Κύριο κλειδί: deptno*

Να κάνετε τις απαραίτητες αλλαγές στο παρακάτω μοντέλο Οντοτήτων Συσχετίσεων.



# Συναρτησιακές εξαρτήσεις



## Μοντέλο Οντοτήτων Συσχετίσεων και Εξαγωγή συναρτησιακών εξαρτήσεων

Μια εταιρεία είναι οργανωμένη σε τμήματα (departments). Κάθε τμήμα έχει ένα μοναδικό όνομα, έναν μοναδικό αριθμό, έναν εργαζόμενο (employee) που το διευθύνει (manages) και έναν αριθμό εργαζομένων που εργάζεται σ' αυτό. Ένα τμήμα ελέγχει (controls) αποκλειστικώς έναν αριθμό έργων (projects) καθένα από τα οποία έχει ένα μοναδικό όνομα, έναν μοναδικό αριθμό και εκτελείται σε μια τοποθεσία. Για κάθε εργαζόμενο κρατούμε: αριθμό ταυτότητας, το πλήρες όνομα (επώνυμο, όνομα, όνομα πατέρα), διεύθυνση, φύλο, μισθό. Κάθε εργαζόμενος ανήκει σε ένα τμήμα αλλά δουλεύει σε διάφορα έργα που δεν ελέγχονται κατ' ανάγκη από το τμήμα του. Για κάθε εργαζόμενο κρατούμε τις ώρες που εργάζεται για κάθε έργο. Για ασφαλιστικούς λόγους κρατούμε τα στοιχεία των μελών της οικογένειας του κάθε εργαζόμενου που είναι εξαρτώμενα από αυτόν: όνομα, φύλο, ημερομηνία γέννησης και σχέση με τον εργαζόμενο.

## Επιχειρησιακός Κανόνας

Κάθε τμήμα έχει ένα μοναδικό όνομα, έναν μοναδικό αριθμό, έναν εργαζόμενο που το διευθύνει.

### ***Συναρτησιακές εξαρτήσεις***

*deptName* → *deptNumber*

*deptNumber* → *deptName*

*deptNumber* → *mngrPidNum*

*deptName* → *mngrPidNum*

## Επιχειρησιακός Κανόνας

Κρατούμε πάντοτε την ημερομηνία που ανέλαβε τη διεύθυνση του τμήματος ο σημερινός διευθυντής, ο οποίος δεν μπορεί να διευθύνει δεύτερο Τμήμα.

Μήπως όμως μας ενδιαφέρει πότε ο σημερινός διευθυντής ανέλαβε το τμήμα για πρώτη φορά;  
Αν μας ενδιαφέρει κάτι τέτοιο (ιστορικά στοιχεία) τότε μας ενδιαφέρει η ΣΕ:

*deptNumber, mngrIdNum, startDate*  $\rightarrow \theta$

## Επιχειρησιακός Κανόνας

Οι δραστηριότητες του τμήματος απλώνονται σε πολλές τοποθεσίες.

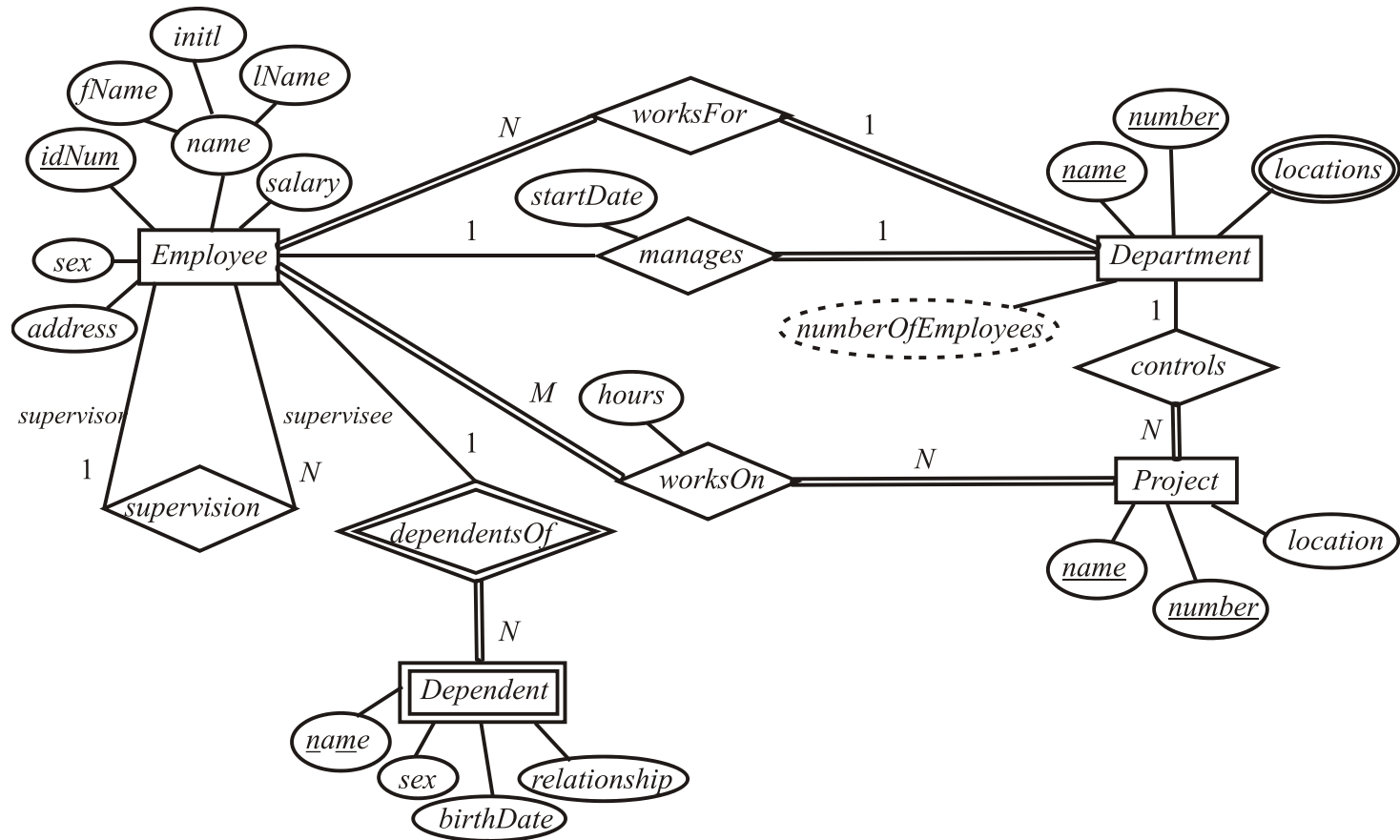
Αυτός ο ΕΚ δεν μας επιτρέπει να γράψουμε κάτι σαν: *deptName* → *deptLocation* αφού αυτό θα σήμαινε ότι το τμήμα είναι εγκατεστημένο σε ένα και μόνο μέρος. Η σωστή ΣΕ είναι:

$$deptNumber, deptLocation \rightarrow \theta$$

$$deptName, deptLocation \rightarrow \theta$$

**Ερώτηση:** Τι θα σήμαινε η *deptLocation* → *deptName*; Θα σήμαινε ότι σε κάθε μέρος υπάρχει ένα μόνο τμήμα της εταιρίας. Π.χ. στην Πανεπιστημίου έχουμε μόνον το Λογιστήριο, στην Ακαδημίας έχουμε τις Πωλήσεις, στην Καλλιθέα τη Διοίκηση, στο Πικέρμι την Παραγωγή (εργοστάσιο), στα Οινόφυτα έχουμε επίσης Παραγωγή (δεύτερο εργοστάσιο) κλπ.

# Παράδειγμα μοντέλου Elmasri – Navathe



Επισκόπηση κάποιων σύνθετων  
εντολών της γλώσσας SQL

## **Απλές αναζητήσεις βασιζόμενες σε ένα πίνακα – Εντολή SELECT ... FROM tablename ...;**

Μία αρκετά γενική μορφή της εντολής αναζήτησης είναι η παρακάτω:

SELECT απλές στήλες, τίτλοι στα αποτελέσματα, πράξεις μεταξύ  
στηλών, συναρτήσεις, τελεστής DISTINCT, χρήση παρενθέσεων  
FROM όνομα πίνακα

WHERE συνθήκη, όπου η συνθήκη ανάλογα και με την εντολή θα  
περιλαμβάνει παρενθέσεις, τελεστές Boole (AND, OR, NOT),  
BETWEEN ... AND, LIKE, τελεστές σύγκρισης (>, <, >=, <=,  
<>), φωλιασμένα αναζήτηση

GROUP BY απλές στήλες ή πράξεις μεταξύ στηλών (επιτρέπεται η  
χρήση παρενθέσεων)

HAVING συνθήκη που περιλαμβάνει στήλες που ανήκουν στην  
υποπρόταση GROUP BY ή άλλες στήλες με ταυτόχρονη χρήση  
(ομαδοποιητικών, αθροιστικών) συναρτήσεων

ORDER BY κριτήρια ταξινόμησης;

Μία κάπως σύνθετη εντολή.

```
SELECT ename, job, sal  
FROM scoEMP  
WHERE (job IN ('ANALYST', 'PROGRAMMER', 'CLERK'))  
AND (sal >= 1300 OR sal+NVL(comm,0) >= 1500)  
AND ename LIKE '%ES'  
AND hiredate BETWEEN '01/01/70' AND '31/12/89'  
ORDER BY job, ename, sal;
```



### Υποαναζήτηση φωλιασμένη σε αναζήτηση

```
SELECT ename, job, sal, deptno  
FROM scoEMP  
WHERE sal > (SELECT MIN(sal)  
             FROM scoEMP  
             WHERE deptno IN (10, 20))  
ORDER BY deptno, ename;
```

Η υποπρόταση SELECT MIN(sal) FROM scoEMP WHERE deptno IN (10, 20)) έχει ως αποτέλεσμα

MIN(SAL)
800

Επομένως η αρχική εκτελείται ως

```
SELECT ename, job, sal, deptno  
FROM scoEMP  
WHERE sal > 800  
ORDER BY deptno, ename;
```

## Χρήση υποπρότασης GROUP BY

```
SELECT deptno, COUNT(*)
```

```
FROM scoemp
```

```
GROUP BY DEPTNO;
```

Ο παρακάτω πίνακας διαμερίζεται λόγω της υποπρότασης GROUP BY DEPTNO

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm.	Deptno
7369	SMITH	CLERK	7902	17/12/00	800		20
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30
7566	JONES	MANAGER	7839	02/04/01	2975		20
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/01	2850		30
7782	CLARK	MANAGER	7839	09/06/01	2450		10
7788	SCOTT	ANALYST	7566	19/04/07	3000		20
7839	KING	PRESIDENT		17/11/01	5000		10
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30
7876	ADAMS	CLERK	7788	23/05/07	1100		20
7900	JAMES	CLERK	7698	03/12/01	950		30
7902	FORD	ANALYST	7566	03/12/01	3000		20
7934	MILLER	CLERK	7782	23/01/02	1300		10

Ως εξής:

7782	CLARK	MANAGER	7839	09/06/01	2450		10
7839	KING	PRESIDENT		17/11/01	5000		10
7934	MILLER	CLERK	7782	23/01/02	1300		10

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm.	Deptno
7369	SMITH	CLERK	7902	17/12/00	800		20
7566	JONES	MANAGER	7839	02/04/01	2975		20
7788	SCOTT	ANALYST	7566	19/04/07	3000		20
7876	ADAMS	CLERK	7788	23/05/07	1100		20
7902	FORD	ANALYST	7566	03/12/01	3000		20

7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/01	2850		30
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30
7900	JAMES	CLERK	7698	03/12/01	950		30

Και το αποτέλεσμα είναι:

deptno	Count (*)
10	3
20	5
30	6

```
SELECT deptno, AVG(sal), COUNT(*)  
FROM scoemp  
GROUP BY DEPTNO;
```

```
SELECT deptno, AVG(sal), COUNT(*)  
FROM scoemp  
GROUP BY DEPTNO  
HAVING deptno > 20 AND COUNT(*)>1;
```

```
SELECT deptno, AVG(sal), COUNT(*)  
FROM scoemp  
GROUP BY DEPTNO  
HAVING (deptno > 20) AND (COUNT(*)>1 OR AVG(sal)>1200);
```

Έστω βάση Διοίκησης Προσωπικού αποτελούμενη από τους παρακάτω πίνακες.

```
SELECT * FROM scoemp;
```

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm.	Deptno
7369	SMITH	CLERK	7902	17/12/00	800		20
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30
7566	JONES	MANAGER	7839	02/04/01	2975		20
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30
7698	BLAKE	MANAGER	7839	01/05/01	2850		30
7782	CLARK	MANAGER	7839	09/06/01	2450		10
7788	SCOTT	ANALYST	7566	19/04/07	3000		20
7839	KING	PRESIDENT		17/11/01	5000		10
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30
7876	ADAMS	CLERK	7788	23/05/07	1100		20
7900	JAMES	CLERK	7698	03/12/01	950		30
7902	FORD	ANALYST	7566	03/12/01	3000		20
7934	MILLER	CLERK	7782	23/01/02	1300		10
7998	BATES	ANALYST	7566	17/11/07	1000		

```
SELECT * FROM scodept;
```

Deptno	Dname	Loc
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

## Απλές αναζητήσεις βασιζόμενες σε περισσότερους από έναν πίνακες

### Εντολή SELECT ... FROM table\_1, table\_2 ... WHERE join ...;

/\* Μπορούμε να γράψουμε απλές εντολές SELECT που θα βασίζονται:

Σε δύο πίνακες (με χρήση σύνδεσης). Σε τρεις ή περισσότερους πίνακες (με χρήση σύνδεσης ανά δύο και λογικών τελεστών). Σε δύο πίνακες (με χρήση εξωτερικής σύνδεσης). \*/

```
SELECT empno, ename, job, sal, sal+nvl(comm,0), scoemp.deptno, dname
```

```
FROM scoemp, scodept
```

```
WHERE scoemp.deptno = scodept.deptno
```

```
AND job IN ('ANALYST', 'PROGRAMMER')
```

```
ORDER BY ename;
```

Σε λογικό επίπεδο γίνεται σύνδεση των δύο πινάκων:

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	scoEmp. Deptno	Dname	Loc
7369	SMITH	CLERK	7902	17/12/00	800		20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30	SALES	CHICAGO
7566	JONES	MANAGER	7839	02/04/01	2975		20	RESEARCH	DALLAS
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	09/06/01	2450		10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	19/04/07	3000		20	RESEARCH	DALLAS
7839	KING	PRESIDENT		17/11/01	5000		10	ACCOUNTING	NEW YORK
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	23/05/07	1100		20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	03/12/01	950		30	SALES	CHICAGO
7902	FORD	ANALYST	7566	03/12/01	3000		20	RESEARCH	DALLAS
7934	MILLER	CLERK	7782	23/01/02	1300		10	ACCOUNTING	NEW YORK
7998	BATES	ANALYST	7566	17/11/07	1000				

Προσοχή η υπαλληλος 7998 δεν συμπεριλαμβάνεται στη σύνδεση.

Και να η επιλογή στηλών (ή πράξεων μεταξύ των στηλών) για προβολή

Empno	Ename	Job	Sal	Sal+NVL (Comm, 0)	scoEmp.Deptno	Dname
7902	FORD	ANALYST	3000	3000	20	RESEARCH
7788	SCOTT	ANALYST	3000	3000	20	RESEARCH

Να και μία παραλλαγή της προηγούμενης εντολής.

```
SELECT empno, ename, job, sal, sal+nvl(comm,0), scoemp.deptno, dname
FROM scodept , scoemp
WHERE scodept.deptno= scoemp.deptno
AND job IN ('ANALYST', 'PROGRAMMER')
ORDER BY ename;
```

Σε λογικό επίπεδο γίνεται σύνδεση των δύο πινάκων:

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	scoEmp. Deptno	Dname	Loc
7782	CLARK	MANAGER	7839	09/06/01	2450		10	ACCOUNTING	NEW YORK
7839	KING	PRESIDENT		17/11/01	5000		10	ACCOUNTING	NEW YORK
7934	MILLER	CLERK	7782	23/01/02	1300		10	ACCOUNTING	NEW YORK
7369	SMITH	CLERK	7902	17/12/00	800		20	RESEARCH	DALLAS
7566	JONES	MANAGER	7839	02/04/01	2975		20	RESEARCH	DALLAS
7788	SCOTT	ANALYST	7566	19/04/07	3000		20	RESEARCH	DALLAS
7876	ADAMS	CLERK	7788	23/05/07	1100		20	RESEARCH	DALLAS
7902	FORD	ANALYST	7566	03/12/01	3000		20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30	SALES	CHICAGO
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	SALES	CHICAGO
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30	SALES	CHICAGO
7900	JAMES	CLERK	7698	03/12/01	950		30	SALES	CHICAGO

Και να η επιλογή στηλών (ή πράξεων μεταξύ των στηλών) για προβολή

Empno	Ename	Job	Sal	Sal+NVL (Comm, 0)	scoEmp.Deptno	Dname
7902	FORD	ANALYST	300 0	3000	20	RESEARCH
7788	SCOTT	ANALYST	300 0	3000	20	RESEARCH



```

SELECT empno, ename, job, sal, sal+nvl(comm,0), scoemp.deptno, dname
FROM scoemp, scodept
WHERE scoemp.deptno = scodept.deptno(+)
AND job IN ('ANALYST', 'PROGRAMMER')
ORDER BY ename;

```

Empno	Ename	Job	Mgr	Hiredate	Sal	Comm	scoEmp. Deptno	Dname	Loc
7369	SMITH	CLERK	7902	17/12/00	800		20	RESEARCH	DALLAS
7499	ALLEN	SALESMAN	7698	20/02/01	1600	300	30	SALES	CHICAGO
7521	WARD	SALESMAN	7698	22/02/01	1250	500	30	SALES	CHICAGO
7566	JONES	MANAGER	7839	02/04/01	2975		20	RESEARCH	DALLAS
7654	MARTIN	SALESMAN	7698	28/09/01	1250	1400	30	SALES	CHICAGO
7698	BLAKE	MANAGER	7839	01/05/01	2850		30	SALES	CHICAGO
7782	CLARK	MANAGER	7839	09/06/01	2450		10	ACCOUNTING	NEW YORK
7788	SCOTT	ANALYST	7566	19/04/07	3000		20	RESEARCH	DALLAS
7839	KING	PRESIDENT		17/11/01	5000		10	ACCOUNTING	NEW YORK
7844	TURNER	SALESMAN	7698	08/09/01	1500	0	30	SALES	CHICAGO
7876	ADAMS	CLERK	7788	23/05/07	1100		20	RESEARCH	DALLAS
7900	JAMES	CLERK	7698	03/12/01	950		30	SALES	CHICAGO
7902	FORD	ANALYST	7566	03/12/01	3000		20	RESEARCH	DALLAS
7934	MILLER	CLERK	7782	23/01/02	1300		10	ACCOUNTING	NEW YORK
7998	BATES	ANALYST	7566	17/11/07	1000				

Προσοχή η υπάλληλος 7998 συμπεριλαμβάνεται στη σύνδεση. Επομένως τα αποτελέσματα είναι:

Empno	Ename	Job	Sal	Sal+NVL (Comm, 0)	scoEmp.Deptno	Dname
7998	BATES	ANALYST	1000	1000		
7902	FORD	ANALYST	3000	3000	20	RESEARCH
7788	SCOTT	ANALYST	3000	3000	20	RESEARCH

## Συνάρτηση DECODE

```
SELECT dname, DECODE(dname, 'ACCOUNTING', 'ΛΟΓΙΣΤΗΡΙΟ',  
                          'RESEARCH', 'ΕΡΕΥΝΑ',  
                          'SALES', 'ΠΩΛΗΣΕΙΣ',  
                          'OPERATIONS', 'ΥΠΟΣΤΗΡΙΞΗ',  
                          'ΆΛΛΟ ΤΜΗΜΑ' )  
  
FROM scoDept;  
SELECT * FROM scoDept;
```

**Η συνάρτηση αυτή μπορεί να μας δώσει πολλές λύσεις και σε πολλά προβλήματα αναδιοργάνωσης της βάσης. Ακολουθεί παράδειγμα δημιουργίας νέου πίνακα και αυτόματη μεταγραφή των στοιχείων από τον παλιό.**

```
CREATE TABLE New_Dept (...);
```

```
INSERT INTO New_Dept
```

```
    SELECT deptno , decode(dname ....), loc FROM scoDept;
```

**Διαφέρουν οι παρακάτω εντολές; Γιατί;**

```
select SUM(sal) / COUNT(sal) from scoEmp;
```

```
select AVG(sal) from scoEmp;
```

```
select SUM(sal) / COUNT(*) from scoEmp;
```

## **Συνάρτηση datediff**

```
SELECT datediff('2015/06/28', current_date);
```

Τι θα δείξει στις 16.6.2015:

**Σύνθεση αναζητήσεων με χρήση τελεστών της θεωρίας συνόλων:**  
**UNION / INTERSECT / MINUS**

Η MySQL δεν έχει τελεστές intersect, minus προς το παρόν.

```
CREATE TABLE staff(ename CHAR(30), hospital CHAR(30));  
CREATE TABLE doctor(ename CHAR(30), hospital CHAR(30));
```

```
INSERT INTO staff VALUES('CODD', 'METAXA');  
INSERT INTO staff VALUES('DATE', 'CARE');  
INSERT INTO staff VALUES('WIDOM', 'METAXA');
```

```
INSERT INTO doctor VALUES('ELMASRI', 'METAXA');  
INSERT INTO doctor VALUES('NAVATHE', 'CARE');  
INSERT INTO doctor VALUES('WIDOM', 'CARE');
```

```
SELECT * FROM staff;  
SELECT * FROM doctor;
```

Η αναζήτηση

```
SELECT ename,hospital
```

```
FROM staff
```

```
WHERE hospital = 'METAXA'
```

```
UNION
```

```
SELECT ename, hospital
```

```
FROM doctor
```

```
WHERE hospital = 'METAXA'
```

## **Να ξαναδούμε κάποιες συναρτήσεις στη συνέχεια**

Η συνάρτηση αυτή μπορεί να μας δώσει πολλές λύσεις και σε πολλά προβλήματα αναδιοργάνωσης της βάσης. Ακολουθεί παράδειγμα δημιουργίας νέου πίνακα και αυτόματη μεταγραφή των στοιχείων από τον παλιό.

```
CREATE TABLE New_Dept (...);  
INSERT INTO New_Dept  
    SELECT deptno , decode(dname ....), loc FROM Dept;
```

**Διαφέρουν οι παρακάτω εντολές; Γιατί;**

```
select SUM(sal) / COUNT(sal) from Emp;  
select AVG(sal) from Emp;  
select SUM(sal) / COUNT(*) from Emp;
```

```

SELECT DNAME ,
        DECODE (RTRIM (DNAME) ,
                'SALES' , 'BARGAIN' ,
                'DEPARTMENTS' )
FROM DEPT;

```

DNAME	DECODE(RTRIM(DNAME),'SALES','BARGAIN','DEPARTMENTS')
ACCOUNTING	DEPARTMENTS
RESEARCH	DEPARTMENTS
SALES	BARGAIN
OPERATIONS	DEPARTMENTS



## Σύγκριση Oracle, MySQL

Oracle	MySQL
<pre>select dname,   <b>DECODE</b> (dname,     'SALES', '1',     'ACCOUNTING', '2',     'Unknown level') FROM dept;</pre>	<pre>select dname, <b>CASE</b> dname   WHEN 'SALES' THEN '1'   WHEN 'ACCOUNTING' THEN '2'   ELSE 'Unknown level' end FROM dept;</pre>

```
mysql> SELECT RTRIM('barbar  ');  
      -> 'barbar'  
mysql> SELECT SUBSTRING('Quadratically',5);  
      -> 'ratically'  
mysql> SELECT SUBSTRING('foobabar' FROM 4);  
      -> 'barbar'  
mysql> SELECT SUBSTRING('Quadratically',5,6);  
      -> 'ratica'  
mysql> SELECT SUBSTRING('Sakila', -3);  
      -> 'ila'  
mysql> SELECT SUBSTRING('Sakila', -5, 3);  
      -> 'aki'  
mysql> SELECT SUBSTRING('Sakila' FROM -4 FOR 2);  
      -> 'ki'
```

## EXISTS - NOT EXISTS

```
SELECT * FROM emp WHERE EXISTS  
  (SELECT * FROM dept WHERE loc='ATHENS' );  
SELECT * FROM emp WHERE NOT EXISTS  
  (SELECT * FROM dept WHERE loc='ATHENS' );  
SELECT * FROM dept WHERE NOT EXISTS  
  (SELECT * FROM emp);
```

```

mysql>
mysql> SELECT * FROM emp WHERE EXISTS
-> (SELECT * FROM dept WHERE loc='ATHENS' );
Empty set (0.00 sec)

```

```

mysql>
mysql> SELECT * FROM emp WHERE NOT EXISTS
-> (SELECT * FROM dept WHERE loc='ATHENS' );

```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	1980-12-17	800.00	NULL	20
7499	ALLEN	SALESMAN	7698	1981-02-20	1600.00	300.00	30
7521	WARD	SALESMAN	7698	2002-02-01	1250.00	500.00	30
7566	JONES	MANAGER	7839	1981-12-24	2975.00	NULL	20
7654	MARTIN	SALESMAN	7698	1981-10-28	1250.00	1400.00	30
7698	BLAKE	MANAGER	7839	2001-05-02	2850.00	NULL	30
7782	CLARK	MANAGER	7839	1981-11-27	2450.00	NULL	10
7788	TT	ANALYST	7566	1987-04-29	3000.00	NULL	20
7839	KING	PRESIDENT	NULL	1987-11-12	5000.00	NULL	10
7844	TURNER	SALESMAN	7698	2007-10-19	1500.00	0.00	30
7876	ADAMS	CLERK	7788	2003-05-07	1100.00	NULL	20
7900	JAMES	CLERK	7698	2003-12-12	950.00	NULL	30
7902	FORD	ANALYST	7566	2003-12-19	3000.00	NULL	20
7934	MILLER	CLERK	7782	2003-01-19	1300.00	NULL	10
7999	BATES	ANALYST	7566	2004-01-04	1300.00	NULL	NULL

```

5 rows in set (0.00 sec)

```

```

mysql>
mysql>
mysql> SELECT * FROM dept WHERE NOT EXISTS
-> (SELECT * FROM emp);
Empty set (0.00 sec)

```

```

mysql>
mysql>

```

```
SELECT * FROM supplier;
```

SUPPL_NAME	SUPPL_CITY
NAVATHE	NEW YORK
ELMASRI	ATHENS

```
SELECT * FROM customer;
```

CUST_NAME	CUST_CITY
CODD	NEW YORK
DATE	BERLIN

Η παρακάτω αναζήτηση βρίσκει ποιοί πελάτες (customers) έχουν έδρα την ίδια πόλη με κάποιο προμηθευτή (supplier):

```
SELECT    cust_name
FROM      customer
WHERE     EXISTS (SELECT suppl_city
                  FROM supplier
                  WHERE supplier.suppl_city = customer.cust_city);
```

CUST_NAME
CODD

## Πώς χρησιμοποιούμε τον τελεστή ANY

**Παράδειγμα** Βρες υπάλληλους που κερδίζουν περισσότερα από (κάποιον) υπάλληλο της Τμήματος 20.

```
SELECT EMPNO, SAL, JOB, ENAME, DEPTNO
FROM EMP
WHERE SAL > ANY
      (SELECT SAL
       FROM EMP
       WHERE DEPTNO = 20)
ORDER BY SAL DESC;
```

```
SELECT EMPNO, SAL, JOB, ENAME, DEPTNO
FROM EMP
WHERE DEPTNO = 20
ORDER BY SAL DESC;
```

```
mysql>
mysql> SELECT EMPNO, SAL, JOB, ENAME, DEPTNO
      -> FROM EMP
      -> WHERE DEPTNO = 20
      -> ORDER BY SAL DESC;
```

EMPNO	SAL	JOB	ENAME	DEPTNO
7788	3000.00	ANALYST	TI	20
7902	3000.00	ANALYST	FORD	20
7566	2975.00	MANAGER	JONES	20
7876	1100.00	CLERK	ADAMS	20
7369	800.00	CLERK	SMITH	20

```
5 rows in set (0.00 sec)
```

```
mysql> SELECT EMPNO,SAL,JOB,ENAME,DEPTNO
-> FROM EMP
-> WHERE SAL > ANY
->      (SELECT SAL
->       FROM EMP
->       WHERE DEPTNO = 20)
-> ORDER BY SAL DESC;
```

EMPNO	SAL	JOB	ENAME	DEPTNO
7839	5000.00	PRESIDENT	KING	10
7902	3000.00	ANALYST	FORD	20
7788	3000.00	ANALYST	IT	20
7566	2975.00	MANAGER	JONES	20
7698	2850.00	MANAGER	BLAKE	30
7782	2450.00	MANAGER	CLARK	10
7499	1600.00	SALESMAN	ALLEN	30
7844	1500.00	SALESMAN	TURNER	30
7934	1300.00	CLERK	MILLER	10
7999	1300.00	ANALYST	BATES	NULL
7654	1250.00	SALESMAN	MARTIN	30
7521	1250.00	SALESMAN	WARD	30
7876	1100.00	CLERK	ADAMS	20
7900	950.00	CLERK	JAMES	30

14 rows in set (0.08 sec)

```
mysql>
```

## Πώς χρησιμοποιούμε τον τελεστή ALL

**Παράδειγμα** Βρες τους υπάλληλους που κερδίζουν περισσότερα  
απο όλους τους υπάλληλους της Τμήματος 20.

```
SELECT * FROM EMP;
```

```
SELECT SAL, JOB, ENAME, DEPTNO  
FROM EMP  
WHERE SAL > ALL  
      (SELECT SAL  
       FROM EMP  
       WHERE DEPTNO = 20)  
ORDER BY SAL DESC;
```



EMPNO	SAL	JOB	ENAME	DEPTNO
7839	5000.00	PRESIDENT	KING	10
7902	3000.00	ANALYST	FORD	20
7788	3000.00	ANALYST	TT	20
7566	2975.00	MANAGER	JONES	20
7698	2850.00	MANAGER	BLAKE	30
7782	2450.00	MANAGER	CLARK	10
7499	1600.00	SALESMAN	ALLEN	30
7844	1500.00	SALESMAN	TURNER	30
7934	1300.00	CLERK	MILLER	10
7999	1300.00	ANALYST	BATES	NULL
7654	1250.00	SALESMAN	MARTIN	30
7521	1250.00	SALESMAN	WARD	30
7876	1100.00	CLERK	ADAMS	20
7900	950.00	CLERK	JAMES	30

14 rows in set (0.00 sec)

```
mysql>
mysql>
mysql>
mysql>
mysql> SELECT SAL,JOB,ENAME,DEPTNO
-> FROM EMP
-> WHERE SAL > ALL
->      (SELECT SAL
->      FROM EMP
->      WHERE DEPTNO = 20)
-> ORDER BY SAL DESC;
```

SAL	JOB	ENAME	DEPTNO
5000.00	PRESIDENT	KING	10

1 row in set (0.00 sec)

mysql>

### Triggers

Έστω η παρακάτω απλοποιημένη βάση δεδομένων.

Πίνακας emp

Ename	Empno	JobNo	Dno
Codd	10	100	10
Elmasri	20	100	10
Navathe	30	200	20
Date	40	300	10

Πίνακας dept

Dno	Abbr	Dname
10	ACCO	ACCOUNTING
20	SALE	SALES
30	PERS	PERSONNEL

Πίνακας job

JobNo	No_of_employees	jobName
100	2	ANALYST
200	1	PROGRAMMER
300	1	SALESMAN
400		DBA

- 1) Δημιουργήστε το σκανδαλισμό (trigger) Insert\_New\_emp έτσι ώστε στην περίπτωση της εισαγωγής των στοιχείων ενός υπαλλήλου στον πίνακα emp να ενεργοποιείται αυτόματα και να προσθέτει 1 μονάδα στη στήλη No\_of\_employees που αντιστοιχεί στη θέση του υπαλλήλου. Επιπλέον, όλα τα στοιχεία του υπαλλήλου θα καταχωρούνται με κεφαλαία.
- 2) Έστω εντολές εισαγωγής στοιχείων τμήματος της μορφής:  
INSERT INTO dept(dno, dname) VALUES (40, 'research');  
Γράψτε το σκανδαλισμό Insert\_New\_dept που θα ενεργοποιείται από την εντολή εισαγωγής και θα καταχωρεί για κάθε τμήμα το όνομα του τμήματος με κεφαλαία στη βάση και θα καταχωρεί, επίσης, συντομογραφία αποτελούμενη από τα 4 πρώτα γράμματα του ονόματος του τμήματος.

### Συναρτησιακές εξαρτήσεις

Έστω ο παρακάτω πίνακας της βάσης δεδομένων βιβλιοπωλείου:

BOOKS (πίνακας στοιχείων βιβλίου)

ISBN	Title	Author (acode, aname)	Publisher (pcode, pname)	Pub_ year	Price
Διεθνής Αριθμός Βιβλίου	Τίτλος	Συγγραφέας (Κωδικός, όνομα)	Εκδότης (Κωδικός,όνομα)	Έτος έκδοσης	Τιμή καταλόγου
0-07-123057-2	Database Management Systems	(100,Ramakrishnan, (200, Gehrke)	10, McGRAW- HILL	2003	70
0-13-727827-6	The essence of databases	(300, Rolland)	20,PRENTICE HALL	1998	30
0-13-861337-0	A first course in database systems	(400, Ullman), (500, Widom)	20,PRENTICE HALL	1997	90

Σημειώστε ποιές είναι σωστές και ποιές λανθασμένες συναρτησιακές εξαρτήσεις:

συναρτησιακές εξαρτήσεις	Σωστό/ Λάθος	Αιτιολογία
1. title --> price	Λ	
2. ISBN, acode --> pub_year	Σ	
3. ISBN, pcode --> pname, pub_year	Σ	
4. ISBN, title, a_code --> price	Σ	
5. acode, pcode --> pub_year	Λ	

Αν  $U$  ένα σύνολο χαρακτηριστικών και  $W, X, Y, Z \subseteq U$  (συχνά αναφέρεται και σαν παγκόσμια σχέση - universal relation) τότε ισχύουν:

α. Τα Αξιώματα του Armstrong (αποδεικνύονται)

1. Αν  $Y \subseteq X$  τότε  $X \twoheadrightarrow Y$
2. Αν  $X \twoheadrightarrow Y$  τότε  $XZ \twoheadrightarrow YZ$
3. Αν  $X \twoheadrightarrow Y$  και  $Y \twoheadrightarrow Z$  τότε  $X \twoheadrightarrow Z$

β. Τα Θεωρήματα

4. Αν  $X \twoheadrightarrow Y$  και  $X \twoheadrightarrow Z$  τότε  $X \twoheadrightarrow YZ$
5. Αν  $X \twoheadrightarrow Y$  και  $WY \twoheadrightarrow Z$  τότε  $WX \twoheadrightarrow Z$
6. Αν  $X \twoheadrightarrow Y$  και  $Z \subseteq Y$  τότε  $X \twoheadrightarrow Z$

Θα αποδείξω τη ΣΕ:  $ISBN, title, a\_code \twoheadrightarrow price$

Ισχύει  $ISBN \twoheadrightarrow price$

Επομένως, από το αξίωμα 2 ισχύει

$ISBN, title, a\_code \twoheadrightarrow title, a\_code, price$

Από το αξίωμα 1 και επειδή ισχύει  $price \subseteq title, a\_code, price$  ισχύει και  $title, a\_code, price \twoheadrightarrow price$

Δηλαδή ισχύουν οι ΣΕ

$ISBN, title, a\_code \twoheadrightarrow title, a\_code, price$

$title, a\_code, price \twoheadrightarrow price$

άρα από το αξίωμα 3 ισχύει και η  $ISBN, title, a\_code \twoheadrightarrow price$

Δηλαδή, αποδείξαμε τη ΣΕ  $ISBN, title, a\_code \twoheadrightarrow price$

Η οποία είναι πλέον περιττή και διαγράφεται.

Πώς αλλιώς θα μπορούσαμε να το αποδείξουμε;

$ISBN \subseteq ISBN, title, a\_code$  άρα  $ISBN, title, a\_code \twoheadrightarrow ISBN$ . Αλλά  $ISBN \twoheadrightarrow price$ . Και τελικά  $ISBN, title, a\_code \twoheadrightarrow price$

πίνακες μίας μη κανονικοποιημένης βάσης  
δεδομένων βιβλιοπωλείου:

BOOKS (πίνακας στοιχείων βιβλίου)

ISBN	Title	Publisher	Publication year	Price	Sales
Διεθνής Αριθμός Βιβλίου	Τίτλος	Εκδότης	Έτος έκδοσης	Τιμή καταλόγου	έκπτωση
0-07-123057-2	Database Management Systems	McGRAW-HILL	2003	70	10
0-13-727827-6	The essence of databases	PRENTICE HALL	1998	30	15
0-13-861337-0	A first course in database systems	PRENTICE HALL	1997	90	30

PUB\_BOOKS\_COUNT (πίνακας στατιστικών που  
αποθηκεύει πόσοι τίτλοι βιβλίων είναι διαθέσιμοι  
ανά εκδότη)

Publisher	No of Books
McGRAW-HILL	1
PRENTICE HALL	2
ACADEMIC PRESS	

Trigger **Insert\_New\_Book\_Trig**. Στην περίπτωση εισαγωγής των στοιχείων βιβλίου στον πίνακα Books ενεργοποιείται αυτόματα και προσθέτει μία μονάδα στη στήλη No\_of\_Books.



Σκανδαλισμός που θα καταχωρεί τους εκδότες με κεφαλαία στη βάση. Ενεργοποιείται από εντολές της μορφής:  
INSERT INTO Books VALUES ('0-13-861337-0 ',  
'A first course in database systems', 'Prentice Hall',1997,90, 30);

Αντί να χρησιμοποιήσετε auto\_increment μπορείτε γράψετε trigger αυτόματης διαχείρισης της τιμής της στήλης Dno. Θα χρειασθεί να ορίσετε μία ενδιάμεση δομή και να αρχικοποιήσετε. Οι τιμές που θα πάρει η στήλη του κωδικού του τμήματος θα μπορούσε να είναι: 1, 2, 3, ... (ή 10, 20, 30, 40, ... κ.λπ.)

```
CREATE TABLE MUSICIAN (M_CODE INT(5), SURNAME VARCHAR(20),  
NAME VARCHAR(15), ADDRESS VARCHAR(50), POSTCODE CHAR(10),  
CITY VARCHAR(20), PHONE_NUMBER VARCHAR(10), PRIMARY KEY(M_CODE));
```

```
CREATE TABLE MAXSEQNO(tablename CHAR(20), MAXACCNO INT);
```

```
Insert into maxseqno VALUES ('MUSICIAN', 0);
```

Βήματα trigger

1) UPDATE MAXSEQNO

SET MAXACCNO=IFNULL(MAXACCNO, 0)+1

WHERE TABLENAME='MUSICIAN';

2) SELECT maxaccno INTO ACCNO\_VAR

FROM maxseqno

WHERE TABLENAME='MUSICIAN';

Κ.λπ.

Δείτε τον παρακάτω πίνακα με στοιχεία αμοιβόμενων εκπαιδευόμενων σπουδαστών

```
mysql> SELECT * FROM STUDENT;
```

student_id	student_name	city	salary	dept_id	dept_name
1	ANN	LONDON	2000.00	1	NULL
2	JOHN	ATHENS	1500.00	1	NULL
3	TOM	ATHENS	3500.00	2	NULL
4	MARY	ATHENS	1700.00	1	NULL
5	PAT	ATHENS	2500.00	2	NULL

```
5 rows in set (0.00 sec)
```

Γράψτε τον bi\_student trigger έτσι ώστε όταν εισάγουμε στοιχεία μισθού να εκτελείται ο trigger και να πολλαπλασιάζει το μισθό με το μήκος του ονόματος.

```
mysql> SELECT * FROM STUDENT;
```

student_id	student_name	city	salary	dept_id	dept_name
1	ANN	LONDON	6000	1	NULL
2	JOHN	ATHENS	6000	1	NULL
3	TOM	ATHENS	10500	2	NULL
4	MARY	ATHENS	6800	1	NULL
5	PAT	ATHENS	7500	2	NULL

```
5 rows in set (0.00 sec)
```

```
DELIMITER //
```

```
create trigger bi_student before insert on student for each row
```

```
Begin
```

```
    declare name_l int;
```

```
    set name_l = length(new.student_name);
```

```
    set new.salary = new.salary * name_l;
```

```
end;
```

```
//
```

```
DELIMITER ;
```

Δείτε τους δύο πίνακες. Γράψτε trigger έτσι ώστε όταν εισάγουμε τα στοιχεία του σπουδαστή να εκτελείται ο trigger και να «διαβάζει» τον πίνακα dept και να εισάγει αυτόματα τιμή στη στήλη dname του πίνακα του σπουδαστή.

```
mysql> Select * From DEPT;
```

dept_id	dept_name
1	COMPUTER SCIENCE
2	INFORMATICS

```
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM STUDENT;
```

student_id	student_name	city	salary	dept_id	dname
1	ANN	LONDON	2000	1	COMPUTER SCIENCE
2	JOHN	ATHENS	1500	1	COMPUTER SCIENCE
3	TOM	ATHENS	3500	2	INFORMATICS
4	MARY	ATHENS	1700	1	COMPUTER SCIENCE
5	PAT	ATHENS	2500	2	INFORMATICS

```
5 rows in set (0.00 sec)
```

DELIMITER //

create trigger bi\_stud\_add\_dname

before insert on student

for each row

begin

declare dnamev varchar(40);

-- μεταβλητή για να τοποθετώ το όνομα του τμήματος που διαβάζω

select dept\_name into dnamev from dept where dept\_id = new.dept\_id;

set new.dname = dnamev; -- έτοιμος να γράψω σωστά το όνομα τμήματος στα στοιχεία

-- σπουδαστή

end;

//

DELIMITER ;

Δείτε τους δύο πίνακες.

```
mysql> Select * From DEPT;
+-----+-----+
| dept_id | dept_name |
+-----+-----+
|      1 | COMPUTER SCIENCE |
|      2 | INFORMATICS      |
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM STUDENT;
+-----+-----+-----+-----+-----+-----+
| student_id | student_name | city   | salary | dept_id | dept_name |
+-----+-----+-----+-----+-----+-----+
|      1    | ANN          | LONDON | 6000   |      1 | COMPUTER SCIENCE |
|      2    | JOHN         | ATHENS | 6000   |      1 | COMPUTER SCIENCE |
|      3    | TOM          | ATHENS | 10500  |      2 | INFORMATICS      |
|      4    | MARY         | ATHENS | 6800   |      1 | COMPUTER SCIENCE |
|      5    | PAT          | ATHENS | 7500   |      2 | INFORMATICS      |
+-----+-----+-----+-----+-----+-----+
5 rows in set (0.00 sec)
```

Δεν μπορούμε μέχρι σήμερα στο προϊόν MySQL να γράψουμε τους 2 παραπάνω triggers που έχουν την ίδια συνθήκη ενεργοποίησης (before insert on student for each row) ώστε να διασμορφώνουμε αυτόματα το περιεχόμενο των στηλών salary, dept\_name του πίνακα student. Δοκιμάστε και θα δείτε το μήνυμα:

```
ERROR 1235 (42000): This version of MySQL doesn't yet support 'multiple triggers with the same action time and event for one table'
```

Γράψτε ένα trigger που θα αναλάβει όλες τις εργασίες που έκαναν οι 2 παραπάνω triggers.

DELIMITER //

create trigger bi\_student before insert on student for each row

Begin

declare name\_length int;

declare dnamev varchar(40);

select dept\_name into dnamev from dept where dept\_id = new.dept\_id;

set new.dept\_name = dnamev;

set name\_length = length(new.student\_name);

set new.salary = new.salary \* name\_length;

end;

//

DELIMITER ;



Γράψτε πρόγραμμα για αυτοματοποίηση λειτουργίας AUDIT για την εισαγωγή των στηλών στον πίνακα student

Drop database school;

CREATE DATABASE school;

USE school;

CREATE TABLE student(student\_id int(7), student\_name varchar(40), city varchar(40), salary int, dept\_id int, dname varchar(40));

CREATE TABLE dept(dept\_id int, dept\_name varchar(40));

DELIMITER //

```
create table audit (user_name varchar(30), table_name varchar(30), update_date date);
create trigger bi_student_audit before insert on student for each row
begin
    insert into audit (user_name, table_name, update_date) values (current_user(),'student',
now());
end;
//
```

DELIMITER ;

Τεστάρισμα

INSERT INTO dept VALUES (1, 'COMPUTER SCIENCE'), (2, 'INFORMATICS');

Select \* From DEPT;

```
INSERT INTO student(student_id, student_name, city, salary, dept_id) VALUES (1, 'ANN', 'LONDON', 2000, 1), (2, 'JOHN', 'ATHENS', 1500, 1), (3, 'TOM', 'ATHENS', 3500, 2), (4, 'MARY', 'ATHENS', 1700, 1), (5, 'PAT', 'ATHENS', 2500, 2);
```

```
SELECT * FROM STUDENT;
```

```
Select * from audit;
```

```
mysql> SELECT * FROM STUDENT;
```

student_id	student_name	city	salary	dept_id	dname
1	ANN	LONDON	2000	1	NULL
2	JOHN	ATHENS	1500	1	NULL
3	TOM	ATHENS	3500	2	NULL
4	MARY	ATHENS	1700	1	NULL
5	PAT	ATHENS	2500	2	NULL

```
5 rows in set (0.00 sec)
```

```
mysql> Select * from audit;
```

user_name	table_name	update_date
root@localhost	student	2012-06-12
root@localhost	student	2012-06-12
root@localhost	student	2012-06-12
root@localhost	student	2012-06-12
root@localhost	student	2012-06-12

```
5 rows in set (0.00 sec)
```

# Procedures - Functions

```
DROP PROCEDURE IF EXISTS balanceCalc;
DELIMITER !
CREATE PROCEDURE balanceCalc ( IN interestRate INT,
                               INOUT balance INT,
                               OUT interest INT)
DETERMINISTIC
BEGIN
SET interest = interestRate * balance / 100;
SET balance = balance + interest;
END !
DELIMITER ;

SET @balance=2000;
SET @interestRate=5;
Select @balance;
CALL balanceCalc(@interestRate, @balance, @interest);
Select @interestRate, @balance, @interest;
```

```
mysql> SET @balance=2000;
Query OK, 0 rows affected (0.00 sec)

mysql> SET @interestRate=5;
Query OK, 0 rows affected (0.00 sec)

mysql> Select @balance;
+-----+
| @balance |
+-----+
|      2000 |
+-----+
1 row in set (0.00 sec)

mysql> CALL balanceCalc(@interestRate, @balance, @interest);
Query OK, 0 rows affected (0.00 sec)

mysql> Select @interestRate, @balance, @interest;
+-----+-----+-----+
| @interestRate | @balance | @interest |
+-----+-----+-----+
|           5 |      2100 |        100 |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> _
```

```

DELIMITER //
CREATE PROCEDURE GetAuthorByCountry(IN countryName VARCHAR(255))
    BEGIN
        SELECT *
        FROM author
        WHERE country = countryName;
    END //
DELIMITER ;

CALL GetAuthorByCountry('GREECE');

DELIMITER $$
CREATE PROCEDURE CountAuthorsByCountry(
    IN AuthorCountry VARCHAR(25),
    OUT total INT)
    BEGIN
        SELECT count(A_ID)
        INTO total
        FROM author
        WHERE country = AuthorCountry;
    END$$
DELIMITER ;

CALL CountAuthorsByCountry('GREECE',@total);

Select @total;

```

```
mysql> SELECT * FROM AUTHOR;
```

A_id	Name	Surname	Country
1	Petros	Belsis	GREECE
2	Nikitas	Karanikolas	GREECE
3	Christos	Skourlas	GREECE
4	Tasos	Tsolakidis	GREECE
5	Dimitris	Vassis	UK

```
5 rows in set (0.00 sec)
```

```
mysql> DELIMITER //
```

```
mysql> CREATE PROCEDURE GetAuthorByCountry(IN countryName VARCHAR(255))  
-> BEGIN  
->     SELECT *  
->     FROM author  
->     WHERE country = countryName;  
-> END //
```

```
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> DELIMITER ;
```

```
mysql> _
```

CALL GetAuthorByCountry('GREECE');

```
MySQL Command Line Client
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 2 to server version: 5.0.27-community-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> USE publications;
Database changed
mysql> CALL GetAuthorByCountry('GREECE');
+-----+-----+-----+-----+
| A_id | Name   | Surname | Country |
+-----+-----+-----+-----+
| 1    | Petros | Belsis  | GREECE  |
| 2    | Nikitas | Karanikolas | GREECE  |
| 3    | Christos | Skourlas | GREECE  |
| 4    | Tasos  | Tsolakidis | GREECE  |
+-----+-----+-----+-----+
4 rows in set (0.06 sec)

Query OK, 0 rows affected (0.08 sec)

mysql>
```



```
mysql> DELIMITER $$
mysql> CREATE PROCEDURE CountAuthorsByCountry(
->     IN AuthorCountry VARCHAR(25),
->     OUT total INT)
-> BEGIN
->     SELECT count(A_ID)
->     INTO total
->     FROM author
->     WHERE country = AuthorCountry;
-> END$$
Query OK, 0 rows affected (0.00 sec)

mysql> DELIMITER ;
mysql>
```

CALL CountAuthorsByCountry('GREECE', @total);  
Select @total;

CALL CountAuthorsByCountry('GREECE', @total);  
Select @total;

```
mysql> CALL CountAuthorsByCountry('GREECE', @total);
Query OK, 0 rows affected (0.00 sec)

mysql> Select @total;
+-----+
| @total |
+-----+
| 4      |
+-----+
1 row in set (0.00 sec)

mysql> _
```

CALL CountAuthorsByCountry('UK', @total);  
Select @total AS TOTAL\_BY\_UK FROM DUAL;

```
mysql>
mysql> CALL CountAuthorsByCountry('UK', @total);
Query OK, 0 rows affected (0.00 sec)

mysql> Select @total AS TOTAL_BY_UK FROM DUAL;
+-----+
| TOTAL_BY_UK |
+-----+
| 1           |
+-----+
1 row in set (0.00 sec)

mysql> _
```

# Υλοποίηση stored procedures: functions και procedures

```
DROP TABLE IF EXISTS myTrace;  
CREATE TABLE myTrace ( t_no INT,  
t_user CHAR(20),  
t_date DATE,  
t_time TIME,  
t_proc VARCHAR(16), t_what VARCHAR(30));  
INSERT INTO myTrace (t_no) VALUES (2);
```

```
mysql>  
mysql>  
mysql> DROP TABLE IF EXISTS myTrace;  
Query OK, 0 rows affected, 1 warning (0.06 sec)  
  
mysql> CREATE TABLE myTrace ( t_no INT,  
-> t_user CHAR(20),  
-> t_date DATE,  
-> t_time TIME,  
-> t_proc VARCHAR(16), t_what VARCHAR(30)  
-> );  
Query OK, 0 rows affected (0.74 sec)  
  
mysql> INSERT INTO myTrace (t_no) VALUES (2);  
Query OK, 1 row affected (0.18 sec)  
  
mysql> SELECT * FROM myTrace;  
+-----+-----+-----+-----+-----+-----+  
| t_no | t_user | t_date | t_time | t_proc | t_what |  
+-----+-----+-----+-----+-----+-----+  
| 2 | NULL | NULL | NULL | NULL | NULL |  
+-----+-----+-----+-----+-----+-----+  
1 row in set (0.00 sec)
```

```
DROP PROCEDURE IF EXISTS myProc;
DELIMITER !
CREATE PROCEDURE myProc (IN p_no INT,IN p_in VARCHAR(30),
        OUT p_out VARCHAR(30))
LANGUAGE SQL
BEGIN
SET p_out = p_in;
INSERT INTO myTrace (t_no, t_user, t_date, t_time, t_proc, t_what)
        VALUES (p_no, current_user, current_date, current_time, 'myProc', p_in);
IF (p_no = 1) THEN
        COMMIT;
ELSE ROLLBACK;
END IF;
END !
DELIMITER ;
```

```
mysql> CALL myProc (1, 'Hello ISO SQL/PSM', @p_out);  
Query OK, 0 rows affected (0.17 sec)
```

```
mysql> Select * from myTrace;
```

t_no	t_user	t_date	t_time	t_proc	t_what
2	NULL	NULL	NULL	NULL	NULL
1	root@localhost	2014-10-28	10:01:20	myProc	Hello ISO SQL/PSM

```
2 rows in set (0.00 sec)
```

# Προσοχή! Δηλώσεις Commit, Rollback δεν επιτρέπονται σε stored functions

-- The mySQL product DOES NOT allow COMMIT and ROLLBACK statements in

-- stored Functions. **Check the following program!**

```
DROP FUNCTION IF EXISTS myFun;
DELIMITER !
CREATE FUNCTION myFun (p_no INT, p_in VARCHAR(30))
RETURNS VARCHAR(30);
LANGUAGE SQL
BEGIN
INSERT INTO myTrace (t_no, t_user, t_date, t_time, t_proc, t_what)
VALUES (p_no, current_user, current_date, current_time, 'myProc', p_in);
IF (p_no = 1) THEN
    COMMIT;
ELSE ROLLBACK;
END IF;
END !
DELIMITER ;
```

# Cursors - Παράδειγμα

**-- Create a function to handle the cursor**

See the following statements:

- Declare variables
- DECLARE CONTINUE HANDLER
- Open Cursor
- Fetch Cursor
- Close Cursor

```
CREATE DATABASE training;
USE training;
CREATE TABLE course(course_id int, course_name varchar(50));
CREATE TABLE lecturer(lecturer_id int(3),
    lecturer_surname varchar(15), lecturer_name varchar(15),
    city varchar(15), salary decimal (8,2), course_id int);
INSERT INTO course VALUES (1, 'DATABASE');
INSERT INTO course VALUES (2, 'WEB DEVELOPMENT');
INSERT INTO course VALUES (3, 'DATA MINING');
INSERT INTO course VALUES (4, 'SEMANTIC WEB');
Select * From COURSE;
INSERT INTO lecturer(lecturer_id, lecturer_name, lecturer_surname,
city, salary, course_id)
VALUES (1, 'CHRIS', 'DATE', 'LONDON', 2000, 1),
(2, 'GIO', 'WIEDERHOLD', 'ATHENS', 1500, 1),
(3, 'PETER', 'CHEN', 'ATHENS', 3500, 2),
(4, 'JEFF', 'ULLMAN', 'ATHENS', 1700, 1),
(5, 'TED', 'CODD', 'ATHENS', 2500, 2);
SELECT lecturer_id, lecturer_surname, lecturer_name, course_id
FROM lecturer;
```



```
mysql> Select * From COURSE;
```

course_id	course_name
1	DATABASE
2	WEB DEVELOPMENT
3	DATA MINING
4	SEMANTIC WEB

4 rows in set (0.00 sec)

```
mysql> SELECT lecturer_id, lecturer_surname, lecturer_name, course_id FROM lecturer;
```

lecturer_id	lecturer_surname	lecturer_name	course_id
1	DATE	CHRIS	1
2	WIEDERHOLD	GIO	1
3	CHEN	PETER	2
4	ULLMAN	JEFF	1
5	CODD	TED	2

5 rows in set (0.00 sec)

```
DELIMITER //
CREATE FUNCTION lecturer_list() RETURNS VARCHAR(255)
BEGIN
DECLARE record_not_found INTEGER DEFAULT 0;
DECLARE lecturer_name_var VARCHAR(150) DEFAULT "";
DECLARE lecturer_surname_var VARCHAR(150) DEFAULT "";
DECLARE lect_list VARCHAR(255) DEFAULT "";
DECLARE my_cursor CURSOR FOR SELECT lecturer_name, lecturer_surname
FROM lecturer;
DECLARE CONTINUE HANDLER FOR NOT FOUND SET record_not_found = 1;
OPEN my_cursor;
    allLecturers: LOOP
        FETCH my_cursor INTO lecturer_name_var, lecturer_surname_var;
        IF record_not_found THEN
            LEAVE allLecturers;
        END IF;
SET lect_list = CONCAT(lect_list, lecturer_surname_var, ", ");
    END LOOP allLecturers;
CLOSE my_cursor;
RETURN SUBSTR(lect_list, 1, 70);
END //
```

```

mysql>
mysql>
mysql> DELIMITER //
mysql> CREATE FUNCTION lecturer_list() RETURNS VARCHAR(255)
      BEGIN
      -> DECLARE record_not_found INTEGER DEFAULT 0;
      -> DECLARE lecturer_name_var VARCHAR(150) DEFAULT "";
      DECLARE lecturer_surname_var VARCHAR(150) DEFAULT "";
      -> DECLARE lect_list VARCHAR(255) DEFAULT "";
      -> DECLARE my_cursor CURSOR FOR SELECT lecturer_name, lecturer_surname FROM
lecturer;
                                     DECLARE CONTINUE HANDLER FOR NOT FOUND SET
record_not_found = 1;          OPEN my_cursor;
      ->      allLecturers: LOOP
      ->          FETCH my_cursor INTO lecturer_name_var, lecturer_surname_var;
      ->          IF record_not_found THEN
      ->              LEAVE allLecturers;
      ->          END IF;
      ->      SET lect_list = CONCAT(lect_list, lecturer_surname_var, ", ");
      ->      END LOOP allLecturers;
      ->      CLOSE my_cursor;
      ->      RETURN SUBSTR(lect_list, 1, 70);
      ->      END //
Query OK, 0 rows affected (0.00 sec)

```

```
DELIMITER ;
```

```
-- Execute function
```

```
mysql> DELIMITER ;  
mysql> SELECT lecturer_list();  
+-----+  
| lecturer_list() |  
+-----+  
| DATE, WIEDERHOLD, CHEN, ULLMAN, CODD, |  
+-----+  
1 row in set (0.00 sec)
```

Ποιά η διαφορά των παρακάτω όψεων αναφορικά με την ενημερωσιμότητα.

```
CREATE VIEW CHECK_SAL  
AS SELECT * FROM emp  
WHERE sal > 1250;
```

```
CREATE VIEW CHECK_SAL  
AS SELECT * FROM emp  
WHERE sal > 1250  
WITH CHECK OPTION;
```

### Πότε έχουμε Updatable και Insertable Views

Αν μπορούμε να χρησιμοποιήσουμε UPDATE, DELETE, or INSERT για να ενημερώσουμε τον πίνακα που «υποκρύπτεται».

Η υποπρόταση WITH CHECK OPTION (clause) μας προστατεύει στην περίπτωση updatable view από inserts, updates που παραβιάζουν την υποπρόταση WHERE (clause) στην εντολή select που χρησιμοποιήσαμε για να ορίσουμε τη view.

Πότε μια view δεν είναι ενημερώσιμη (is not updatable):

Όταν περιλαμβάνει

1. Aggregate functions (SUM(), MIN(), MAX(), COUNT(), ...)
2. DISTINCT
3. GROUP BY
4. HAVING
5. UNION
6. Subquery (πράξεις κ.λπ.) in the select list
7. Certain joins
8. Nonupdatable view in the FROM clause
9. A subquery in the WHERE clause that refers to a table in the FROM clause

## DBA (Oracle)

Ακολουθεί μία περιγραφή καθηκόντων του μέσα απο παραδείγματα .

```
CONNECT SYSTEM/(password)
```

```
GRANT CONNECT TO FORD IDENTIFIED BY CAR;
```

```
CONNECT FORD/CAR
```

```
CONNECT SYSTEM/(password)
```

```
GRANT RESOURCE TO FORD;
```

```
CONNECT FORD/CAR
```

```
CREATE TABLE PARTS (PARTNO NUMBER,
```

```
DESCRIPTION CHAR(20),QUANTITY NUMBER);
```

```
GRANT SELECT ON PARTS TO PUBLIC;
```

```
CONNECT SYSTEM/(password)
```

```
GRANT DBA TO KING IDENTIFIED BY ALEXANDER;
```



Πώς ένας υπάλληλος βλέπει μόνο τα στοιχεία που τον αφορούν.

Δημιουργούμε κατάλληλα την όψη

```
CREATE VIEW MYSELF
```

```
AS SELECT * FROM EMP WHERE ENAME = USER;
```

και δίνουμε δικαίωμα αναζήτησης σε όλους (GRANT ... TO PUBLIC)

```
GRANT SELECT ON MYSELF TO PUBLIC;
```

Τότε ο οποιοσδήποτε (π.χ. ο ADAMS) μπορεί να βλέπει μόνο τα στοιχεία του.

```
CONNECT ADAMS/JIM
```

```
SELECT * FROM SCOURLAS.MYSELF;
```

Προσοχή το προϊόν της mysql λειτουργεί όταν συνδεθούμε σε AUTOCOMMIT mode.

```
mysql> DESCRIBE T;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| id    | int(11)       | NO   | PRI | NULL    |       |
| s     | varchar(30)   | YES  |     | NULL    |       |
| si    | smallint(6)   | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

mysql> INSERT INTO T (id, s) VALUES (1, 'first');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO T (id, s) VALUES (2, 'second');
Query OK, 1 row affected (0.05 sec)

mysql> INSERT INTO T (id, s) VALUES (3, 'third');
Query OK, 1 row affected (0.06 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s       | si |
+-----+-----+-----+
| 1  | first  | NULL |
| 2  | second | NULL |
| 3  | third  | NULL |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s       | si |
+-----+-----+-----+
| 1  | first  | NULL |
| 2  | second | NULL |
| 3  | third  | NULL |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql>
```

Να πως ξεκινάμε και εκτελούμε transaction

```
mysql> START TRANSACTION;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO T (id, s) VALUES (4, 'fourth');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM T ;
+----+-----+-----+
| id | s      | si    |
+----+-----+-----+
| 1  | first  | NULL  |
| 2  | second | NULL  |
| 3  | third  | NULL  |
| 4  | fourth | NULL  |
+----+-----+-----+
4 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.05 sec)

mysql> SELECT * FROM T;
+----+-----+-----+
| id | s      | si    |
+----+-----+-----+
| 1  | first  | NULL  |
| 2  | second | NULL  |
| 3  | third  | NULL  |
+----+-----+-----+
3 rows in set (0.00 sec)
```

```
START TRANSACTION;
INSERT INTO T (id, s) VALUES (4, 'fourth');
SELECT * FROM T ;
ROLLBACK;
SELECT * FROM T;
```

Προσοχή! Με την εντολή ROLLBACK ολοκληρώθηκε η TRANSACTION και το προϊόν της mySQL λειτουργεί και πάλι σε AUTOCOMMIT mode.

## Switch off the AUTOCOMMIT mode

```
SET AUTOCOMMIT=0;
```

Τώρα λειτουργεί σωστά κάθε transaction όπως μπορείτε να διαπιστώσετε.

```
INSERT INTO T (id, s) VALUES (2, 'second');  
INSERT INTO T (id, s) VALUES (3, 'third');  
SELECT * FROM T;  
ROLLBACK;  
SELECT * FROM T;  
INSERT INTO T (id, s) VALUES (5, 'fifth');  
ROLLBACK;  
SELECT * FROM T;
```

κάθε transaction ολοκληρώνεται με COMMIT ή ROLLBACK.

Είναι γνωστές οι εντολές της Data Definition Language (DDL): “CREATE TABLE...”, “CREATE INDEX...”, “DROP TABLE ...” κ.λπ.

Είναι γνωστές επίσης οι εντολές της Data Manipulation Language (DML): “SELECT FROM...”, “INSERT INTO...”, “DELETE FROM...”. Επηρεάζουν αυτές οι εντολές την εντολή ROLLBACK;

```
INSERT INTO T (id, s) VALUES (9, 'will this be committed?');
CREATE TABLE T2 (id INT);
INSERT INTO T2 (id) VALUES (1);
SELECT * FROM T2;
ROLLBACK;
SELECT * FROM T; -- What has happened to T ?
SELECT * FROM T2; -- What has happened to T2 ?
SELECT * FROM T3; -- Oops!
SHOW TABLES;
```

```
mysql> INSERT INTO T (id, s) VALUES (9, 'will this be committed?');
Query OK, 1 row affected (0.02 sec)

mysql> CREATE TABLE T2 (id INT);
Query OK, 0 rows affected (0.14 sec)

mysql> INSERT INTO T2 (id) VALUES (1);
Query OK, 1 row affected (0.02 sec)

mysql> SELECT * FROM T2;
+-----+
| id    |
+-----+
| 1     |
+-----+
1 row in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)

mysql> SELECT * FROM T; -- What has happened to T ?
+-----+-----+-----+
| id | s                | si |
+-----+-----+-----+
| 1  | first            | NULL |
| 9  | will this be committed? | NULL |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> SELECT * FROM T2; -- What has happened to T2 ?
Empty set (0.01 sec)

mysql> SELECT * FROM T3; -- Oops!
ERROR 1146 (42S02): Table 'testdb.t3' doesn't exist
mysql> SHOW TABLES;
+-----+
| Tables_in_testdb |
+-----+
| t                 |
| t2                |
+-----+
2 rows in set (0.03 sec)
```

```
INSERT INTO T (id, s) VALUES (2, 'Error test starts here');
-- division by zero should fail
SELECT (1/0) AS dummy FROM DUAL;
-- Now update a non-existing row
UPDATE T SET s = 'foo' WHERE id = 9999 ;
-- and delete an non-existing row
DELETE FROM T WHERE id = 7777 ;
--
INSERT INTO T (id, s) VALUES (2, 'Hi, I am a duplicate');
INSERT INTO T (id, s)
VALUES (3, 'How about inserting too long of a string value?');
INSERT INTO T (id, s, si) VALUES (4, 'Smallint overflow for 32769?', 32769);
INSERT INTO T (id, s) VALUES (5, 'Transaction still active?');
SELECT * FROM T;
COMMIT;
DELETE FROM T WHERE id > 1;
SELECT * FROM T;
COMMIT;
```

ΠΡΟΣΟΧΗ! Η τιμή σφάλματος “23000” είναι η τιμή του SQLSTATE που ορίζεται στο standard και η τιμή 1062 είναι η τιμή του αντίστοιχου κώδικα σφάλματος του προϊόντος της MySQL. Συνδέεται με παραβίαση περιορισμού κύριου κλειδιού (violation of the primary key constraint).

Να κάνετε έλεγχο αν ένα σφάλμα (error) σε εντολή SQL οδηγεί ή όχι σε αυτόματο ROLLBACK στο προϊόν της MySQL. Εκτελέστε τις εντολές στη συνέχεια.

```
mysql> INSERT INTO T (id, s) VALUES (2, 'Error test starts here');
Query OK, 1 row affected (0.02 sec)

mysql> -- division by zero should fail
mysql> SELECT (1/0) AS dummy FROM DUAL;
+-----+
| dummy |
+-----+
| NULL  |
+-----+
1 row in set (0.05 sec)

mysql> -- Now update a non-existing row
mysql> UPDATE T SET s = 'foo' WHERE id = 9999 ;
Query OK, 0 rows affected (0.01 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> -- and delete an non-existing row
mysql> DELETE FROM T WHERE id = 7777 ;
Query OK, 0 rows affected (0.00 sec)

mysql> --
mysql> INSERT INTO T (id, s) VALUES (2, 'Hi, I am a duplicate');
ERROR 1062 (23000): Duplicate entry '2' for key 'PRIMARY'
mysql> INSERT INTO T (id, s)
-> VALUES (3, 'How about inserting too long of a string value?');
ERROR 1406 (22001): Data too long for column 's' at row 1
mysql> INSERT INTO T (id, s, si) VALUES (4, 'Smallint overflow for 32769?', 32769);
ERROR 1264 (22003): Out of range value for column 'si' at row 1
mysql> INSERT INTO T (id, s) VALUES (5, 'Transaction still active?');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s                | si |
+-----+-----+-----+
| 1 | first            | NULL |
| 2 | Error test starts here | NULL |
| 5 | Transaction still active? | NULL |
+-----+-----+-----+
3 rows in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.03 sec)

mysql> DELETE FROM T WHERE id > 1;
Query OK, 2 rows affected (0.00 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s      | si |
+-----+-----+-----+
| 1 | first | NULL |
+-----+-----+-----+
1 row in set (0.00 sec)
```

Το προϊόν της MySQL δεν υποστηρίζει CHECK constraints

```
CREATE TABLE Accounts (  
  acctID INTEGER NOT NULL PRIMARY KEY,  
  balance INTEGER NOT NULL,  
  CONSTRAINT unloanable_account CHECK (balance >= 0)  
);  
INSERT INTO Accounts (acctID,balance) VALUES (101,1000);  
INSERT INTO Accounts (acctID,balance) VALUES (202,2000);  
SELECT * FROM Accounts;  
COMMIT;  
-- let's try the bank transfer  
UPDATE Accounts SET balance = balance - 100 WHERE acctID = 101;  
UPDATE Accounts SET balance = balance + 100 WHERE acctID = 202;  
SELECT * FROM Accounts;  
ROLLBACK;  
-- Let's test the CHECK constraint actually work:  
UPDATE Accounts SET balance = balance - 2000 WHERE acctID = 101;  
UPDATE Accounts SET balance = balance + 2000 WHERE acctID = 202;  
SELECT * FROM Accounts ;  
ROLLBACK;
```



Δοκιμάστε SQL transaction για τη μεταφορά 500 euros από το λογαριασμό 101 σε ανύπαρκτο λογαριασμό πχ στον acctID=777:

UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;

UPDATE Accounts SET balance = balance + 500 WHERE acctID = 777;

SELECT \* FROM Accounts ;

ROLLBACK;

```
mysql> UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Accounts SET balance = balance + 500 WHERE acctID = 777;
Query OK, 0 rows affected (0.00 sec)
Rows matched: 0  Changed: 0  Warnings: 0

mysql> SELECT * FROM Accounts ;
+-----+-----+
| acctID | balance |
+-----+-----+
|    101 |    500 |
|    202 |   2000 |
+-----+-----+
2 rows in set (0.00 sec)

mysql> ROLLBACK;
Query OK, 0 rows affected (0.03 sec)
```

## SQL Transaction: Unit of Recovery

```
mysql>
mysql> SET AUTOCOMMIT = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO T (id, s) VALUES (9, 'Let''s see what happens if ..');
Query OK, 1 row affected (0.00 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s | si |
+-----+-----+-----+
| 1 | first | NULL |
| 9 | Let's see what happens if .. | NULL |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql>
```

Κλείστε βίαια το (client) session για να προσομοιώσετε μία κατάσταση “DBMS crash”.

Ανοίξτε ένα νέο terminal window (νέα MySQL session) και δείτε τι έγινε.

```
USE TestDB;
SET AUTOCOMMIT=0;
SELECT * FROM T;
COMMIT;
```

```
mysql> USE TestDB;
Database changed
mysql> SET AUTOCOMMIT=0;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT * FROM T;
+-----+-----+-----+
| id | s | si |
+-----+-----+-----+
| 1 | first | NULL |
+-----+-----+-----+
1 row in set (0.00 sec)

mysql> COMMIT;
Query OK, 0 rows affected (0.00 sec)
```

Work with table T:

```
use TestDB
SET AUTOCOMMIT=0;
```

```
DROP TABLE T;
CREATE TABLE T (id INT NOT NULL PRIMARY KEY, s VARCHAR(30), si SMALLINT);
INSERT INTO T(id,s) VALUES (1,'first'), (2,'second'), (3,'third');
COMMIT;
```

### **SIMULATE A DEADLOCK SITUATION:**

=====

Step	Session A	Session B
1	UPDATE T SET si=333 WHERE id=1;	
2		UPDATE T SET si=222 WHERE id=2;
3	SELECT * FROM T;	
4		SELECT * FROM T;
5	UPDATE T SET si=22 WHERE id=2;	
6		UPDATE T SET si=333 WHERE id=2;
7	COMMIT;	
8		COMMIT;

Αν θέλουμε να μεταφέρουμε χρήματα απο ένα λογαριασμό σε άλλο χρειαζόμαστε δύο απλές κινήσεις :

```
UPDATE ACCOUNTS  
SET BALANCE = BALANCE - 100000  
WHERE ACCNO = 120768;
```

```
UPDATE ACCOUNTS  
SET BALANCE = BALANCE + 100000  
WHERE ACCNO = 345678;
```

**Πως θα εξασφαλίσουμε την ασφάλεια της συναλλαγής;**

## Transaction Logic: Problems related to a simple bank example

```
-- Transferring of 100 euros from one account to another
CREATE TABLE Accounts (
    acctId INTEGER NOT NULL PRIMARY KEY,
    balance DECIMAL(11,2) CHECK (balance >= 0.00));
BEGIN TRANSACTION;
UPDATE Accounts SET balance = balance - 100
WHERE acctId = 101;
UPDATE Accounts SET balance = balance + 100
WHERE acctId = 202;
COMMIT;
```

### Example of a solution in mySQL

```
DELIMITER //
DROP PROCEDURE BankTransfer //
CREATE PROCEDURE BankTransfer (IN fromAcct INT,
                               IN toAcct INT,
                               IN amount INT,
                               OUT msg VARCHAR(100)
                              )
P1: BEGIN
    DECLARE rows INT ;
    DECLARE newbalance INT;
    SELECT COUNT(*) INTO rows FROM Accounts WHERE acctID = fromAcct;
    UPDATE Accounts SET balance = balance - amount WHERE acctID = fromAcct;
    SELECT balance INTO newbalance FROM Accounts WHERE acctID =
fromAcct;
    IF rows = 0 THEN
        ROLLBACK;
        SET msg = CONCAT('rolled back because of missing account ', fromAcct);
    ELSEIF newbalance < 0 THEN
        ROLLBACK;
        SET msg = CONCAT('rolled back because of negative balance of account ',
fromAcct);
    ELSE
        SELECT COUNT(*) INTO rows FROM Accounts WHERE acctID = toAcct;
        UPDATE Accounts SET balance = balance + amount WHERE acctID =
toAcct;
        IF rows = 0 THEN
            ROLLBACK;
            SET msg = CONCAT('rolled back because of missing account ', toAcct);
        ELSE
            COMMIT;
            SET msg = 'committed';
        END IF;
    END IF;
END P1 //
DELIMITER ;
```

```
mysql> SET @out = '';
Query OK, 0 rows affected (0.00 sec)
mysql> CALL BankTransfer (101, 202, 100, @out);
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT @out;
```

```
+-----+
```

```
| @out      |
```

```
+-----+
```

```
| committed |
```

```
+-----+
```

```
1 row in set (0.00 sec)
```

```
CALL BankTransfer (100, 202, 100, @out);
CALL BankTransfer (101, 200, 100, @out);
CALL BankTransfer (101, 202, 2000, @out);
```

```
DROP DATABASE TestDB;
```

```
CREATE DATABASE TestDB;
```

```
use TestDB;
```

```
CREATE TABLE Accounts (
```

```
acctID INTEGER NOT NULL PRIMARY KEY,
```

```
cname CHAR(20), balance INTEGER NOT NULL,
```

```
CONSTRAINT unloanable_account CHECK (balance >= 0)
```

```
);
```

```
INSERT INTO Accounts (acctID, cname, balance)
```

```
VALUES (101, 'ARTHUR', 1000);
```

```
INSERT INTO Accounts (acctID,cname, balance)
```

```
VALUES (202, 'JIM', 2000);
```

```
INSERT INTO Accounts (acctID,cname, balance)
```

```
VALUES (303, 'TOM', 2000);
```

```
SELECT * FROM Accounts;
```



```
mysql> use TestDB;
Database changed
mysql> CREATE TABLE Accounts (
    -> acctID INTEGER NOT NULL PRIMARY KEY,
    -> cname CHAR(20), balance INTEGER NOT NULL,
    -> CONSTRAINT unloanable_account CHECK (balance >= 0)
    -> );
```

Query OK, 0 rows affected (0.09 sec)

```
mysql> INSERT INTO Accounts (acctID, cname, balance)
    -> VALUES (101, 'ARTHUR', 1000);
```

Query OK, 1 row affected (0.05 sec)

```
mysql> INSERT INTO Accounts (acctID, cname, balance)
    -> VALUES (202, 'JIM', 2000);
```

Query OK, 1 row affected (0.06 sec)

```
mysql> INSERT INTO Accounts (acctID, cname, balance)
    -> VALUES (303, 'TOM', 2000);
```

Query OK, 1 row affected (0.03 sec)

```
mysql> SELECT * FROM Accounts;
```

acctID	cname	balance
101	ARTHUR	1000
202	JIM	2000
303	TOM	2000

3 rows in set (0.00 sec)

```
mysql>
```

## Non repeatable Read in mySQL

Step	Session A	Session B
1	SET AUTOCOMMIT = 0; SET TRANSACTION ISOLATION LEVEL READ COMMITTED; SELECT * FROM Accounts WHERE balance > 500;	
2		SET AUTOCOMMIT = 0; SET TRANSACTION ISOLATION LEVEL REPEATABLE READ; UPDATE Accounts SET balance = balance - 500 WHERE acctID = 101; UPDATE Accounts SET balance = balance + 500 WHERE acctID = 202; COMMIT WORK;
3	-- repeating the same query SELECT * FROM Accounts WHERE balance > 500; COMMIT;	

# Session A

```
mysql> SET AUTOCOMMIT = 0;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> SELECT *  
      -> FROM Accounts  
      -> WHERE balance > 500;
```

acctID	cname	balance
101	ARTHUR	1000
202	JIM	2000
303	TOM	2000

```
3 rows in set (0.00 sec)
```

```
mysql>
```

# Session B

```
mysql> USE testDB;
Database changed
mysql> SET AUTOCOMMIT = 0;
Query OK, 0 rows affected (0.00 sec)

mysql> SET TRANSACTION ISOLATION LEVEL REPEATABLE READ;
Query OK, 0 rows affected (0.00 sec)

mysql> UPDATE Accounts
    -> SET balance = balance - 500
    -> WHERE acctID = 101;
Query OK, 1 row affected (0.09 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> UPDATE Accounts
    -> SET balance = balance + 500
    -> WHERE acctID = 202;
Query OK, 1 row affected (0.02 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> COMMIT WORK;
Query OK, 0 rows affected (0.05 sec)

mysql> _
```

# Session A

```
mysql> SELECT *  
      -> FROM Accounts  
      -> WHERE balance > 500;
```

acctID	cname	balance
101	ARTHUR	1000
202	JIM	2000
303	TOM	2000

```
3 rows in set (0.00 sec)
```

```
mysql> -- repeating the same query
```

```
mysql> SELECT *  
      -> FROM Accounts  
      -> WHERE balance > 500;
```

acctID	cname	balance
202	JIM	2500
303	TOM	2000

```
2 rows in set (0.02 sec)
```

```
mysql> COMMIT;
```

```
Query OK, 0 rows affected (0.00 sec)
```

# Blind Overwriting problem, application simulated by use of local variables

Step	Session A	Session B
1	<pre>SET AUTOCOMMIT=0;  SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  -- amount to be transfered by A  SET @amountA = 200;  SET @balanceA = 0; -- init value  SELECT balance INTO @balanceA  FROM Accounts  WHERE acctID = 101;  SET @balanceA = @balanceA - @amountA;  SELECT @balanceA;</pre>	<pre>mysql&gt; SELECT @balanceA; +-----+   @balanceA   +-----+            800   +-----+ 1 row in set (0.02 sec)</pre>

Step	Session A	Session B
2		SET AUTOCOMMIT=0;  SET TRANSACTION ISOLATION LEVEL  READ COMMITTED;  -- amount to be transfered by B  SET @amountB = 500;  SET @balanceB = 0; -- init value  SELECT balance INTO @balanceB  FROM Accounts WHERE acctID = 101;  SET @balanceB = @balanceB - @amountB;  SELECT @balanceB;

```
mysql> SELECT @balanceB;
+-----+
| @balanceB |
+-----+
|          500 |
+-----+
1 row in set (0.00 sec)
```

Step	Session A	Session B
3	UPDATE Accounts SET balance = @balanceA WHERE acctID = 101;	
4		UPDATE Accounts  SET balance = @balanceB  WHERE acctID = 101;
5	SELECT acctID, balance FROM Accounts WHERE acctID = 101; COMMIT;	

```
mysql> SELECT acctID, balance
-> FROM Accounts
-> WHERE acctID = 101;
```

```
+-----+-----+
| acctID | balance |
+-----+-----+
|    101 |    800 |
+-----+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.05 sec)
```



Step	Session A	Session B
6		SELECT acctID, balance FROM Accounts WHERE acctID = 101; COMMIT;

```
mysql> SELECT acctID, balance
-> FROM Accounts
-> WHERE acctID = 101;
```

```
+-----+-----+
| acctID | balance |
+-----+-----+
|    101 |    500  |
+-----+-----+
```

```
1 row in set (0.00 sec)
```

```
mysql> COMMIT;
```

```
Query OK, 0 rows affected (0.03 sec)
```

7	SELECT * FROM Accounts;	
8		SELECT * FROM Accounts;

```
mysql> SELECT * FROM Accounts;
+-----+-----+
| acctID | balance |
+-----+-----+
|    101 |    500  |
|    202 |   2000  |
+-----+-----+
2 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM Accounts;
+-----+-----+
| acctID | balance |
+-----+-----+
|    101 |    500  |
|    202 |   2000  |
+-----+-----+
2 rows in set (0.00 sec)
```

Αναφέρατε βασικές πράξεις της Σχεσιακής άλγεβρας

FOITHTES

<u>ARITMHT</u>	<u>EPWNYMO</u>	<u>ONOMA</u>	<u>EXAMHNO</u>
10	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	1
20	ΝΙΚΟΥ	ΝΙΚΟΣ	4

EGGRAFES

<u>ARITMHT</u>	<u>LESSON</u>	<u>HMER EGGRAPHIS</u>
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2011
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2011

1. Επιλογή (Selection):  $R_1 = \sigma_E(R)$

Το αποτέλεσμα της πράξης αυτής, που συμβολίζεται με  $\sigma$  και εφαρμόζεται πάνω σε μία σχέση  $R$ , είναι μια σχέση  $R_1$  που έχει την ίδια πολλαπλότητα με την  $R$  και περιλαμβάνει τις πλειάδες της  $R$  για τις οποίες ισχύει η συνθήκη  $E$ .

Παράδειγμα

Η  $FOITHTES1 = \sigma_{EXAMHNO=1}(FOITHTES)$  περιέχει μόνο εκείνες τις πλειάδες της σχέσης  $R$  όπου το  $EXAMHNO$  έχει την τιμή 1.

## Αναφέρατε βασικές πράξεις της Σχεσιακής άλγεβρας

FOITHTES

<u>ARITMHT</u>	<u>EPWNYMO</u>	<u>ONOMA</u>	<u>EXAMHNO</u>
10	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	Γ
20	ΝΙΚΟΥ	ΝΙΚΟΣ	Δ

EGGRAFES

<u>ARITMHT</u>	<u>LESSON</u>	<u>HMER_EGGRAPHIS</u>
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2011
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ Ι	10/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ ΙΙ	9/3/2011

2. Προβολή (projection):  $R_1 = \pi_{a_1, a_2, a_3, \dots, a_k}(R)$

Αν η  $R$  είναι μια σχέση πολλαπλότητας  $N$  η  $R_1$  είναι μια σχέση πολλαπλότητας  $k < N$  με χαρακτηριστικά  $a_1, a_2, \dots, a_k$ , όπου τα  $a_m$  είναι χαρακτηριστικά της  $R$  και το  $m$  μπορεί να έχει τιμές από 1 μέχρι  $k$ . Δηλαδή, η προβολή εφαρμόζεται σε μία σχέση και δείχνει τη στήλη ή τις στήλες της σχέσης (ή πίνακα) που θέλουμε.

Παράδειγμα

$FOITHTES2 = \pi_{ARITMHT, EXAMHNO, EPWNYMO}(FOITHTES)$

### Αναφέρατε βασικές πράξεις της Σχεσιακής άλγεβρας

FOITHTES

<u>ARITMHT</u>	<u>EPWNYMO</u>	<u>ONOMA</u>	<u>EXAMHNO</u>
10	ΣΠΥΡΟΥ	ΣΠΥΡΟΣ	1
20	ΝΙΚΟΥ	ΝΙΚΟΣ	4

EGGRAFES

<u>ARITMHT</u>	<u>LESSON</u>	<u>HMER EGGRAPHIS</u>
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ I	10/3/2011
10	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II	9/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ I	10/3/2011
20	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ II	9/3/2011

### **3. Καρτεσιανό γινόμενο (Cartesian Product or Times)**

Αν η σχέση  $R_1$  έχει σχήμα  $A_1, A_2, \dots, A_v$  και η σχέση  $R_2$  έχει σχήμα  $B_1, B_2, \dots, B_\mu$  τότε το καρτεσιανό γινόμενο των δύο σχέσεων, συμβολικά  $R_1 \times R_2$ , ορίζεται σαν μία σχέση με σχήμα  $(A_1, A_2, \dots, A_v, B_1, B_2, \dots, B_\mu)$ , δηλαδή,

$$R_1 \times R_2 = \{ (a_1, \dots, a_v, \beta_1, \dots, \beta_\mu) \mid (a_1, \dots, a_v) \in R_1, \\ (\beta_1, \dots, \beta_\mu) \in R_2 \}$$

Αν  $\text{card}(R)$  είναι ο αριθμός των πλειάδων (tuples) μιας σχέσης  $R$  τότε

$$\text{card}(R_1 \times R_2) \leq \text{card}(R_1) \times \text{card}(R_2)$$

Όπως είναι προφανές η έννοια γενικεύεται άμεσα για περισσότερες σχέσεις.

---

#### 4. Σύνδεση (join) σχέσεων : $R = R_1 \bowtie R_2$

Αν  $R_1, R_2$  είναι σχέσεις με πολλαπλότητες  $N1$  και  $N2$  και με  $M$  κοινά χαρακτηριστικά των δύο σχέσεων κατασκευάζουμε την  $R$  που είναι μια σχέση με πολλαπλότητα

$$N = N1 + N2 - M.$$

Η  $R$  δημιουργείται με την παράθεση (concatenation) όλων των πλειάδων  $t_1 \in R_1$  και  $t_2 \in R_2$  που ικανοποιούν τη συνθήκη της σύνδεσης.

Σχηματικά, για όλες τις πλειάδες των δύο σχέσεων κάνουμε τα εξής:

i) Συνδέουμε κάθε πλειάδα της  $R_1$  με όλες τις πλειάδες της  $R_2$  και δημιουργούμε έτσι πλειάδες με πολλαπλότητα  $N1 + N2$ .

ii) Από αυτές κρατάμε μόνον αυτές που έχουν σε όλα τα κοινά χαρακτηριστικά τις ίδιες τιμές.

iii) Σε αυτές που κρατήσαμε, κρατάμε τα (κοινά) χαρακτηριστικά μια φορά μόνον.

Παράδειγμα

$$EGGRFOIT = FOITHTES \bowtie EGGRAFES$$

Παράδειγμα

Retrieve the name and address of all employees who work for the “Development” department

EMP

<u>EMPNO</u>	<u>ENAME</u>	<u>ADDRESS</u>	<u>DEPTNO</u>
10	CLARKE	ATHENS	10
20	ADAMS	NEW YORK	20

DEPT

<u>DNO</u>	<u>DNAME</u>
10	DEVELOPMENT
20	RESEARCH

$\text{DEVELOPMENT\_DEPT} \leftarrow \sigma_{\text{dname} = \text{"DEVELOPMENT"}} (\text{DEPT})$

$\text{DEVELOPMENT\_EMP} \leftarrow (\text{DEVELOPMENT\_DEPT} \bowtie_{\text{DNO} = \text{DEPTNO}} \text{EMP})$

$\text{RESULT} \leftarrow \pi_{\text{ENAME}, \text{ADDRESS}} (\text{DEVELOPMENT\_EMP})$

Εναλλακτικά join ανάμεσα στις σχέσεις EMP, DEPT και στη συνέχεια projection. Προσοχή!  
Η εκτέλεση του query είναι πιο χρονοβόρα.

### Κανονική μορφή BOYCE-CODD

Έστω βάση δεδομένων εκπαιδευτικού ιδρύματος. Δώστε παράδειγμα πίνακα που είναι στην Τρίτη κανονική μορφή αλλά όχι στην κανονική μορφή Boyce-Codd. Μη ξεχάσετε να αναφέρετε τους περιορισμούς που ισχύουν. Γράψτε την κανονική μορφή Boyce-Codd. Τι κερδίσαμε πηγαίνοντας στη μορφή αυτή;

Έστω ότι ο πίνακας STUD\_COUR\_TEACH ανήκει στο Σύστημα Βάσης Δεδομένων ενός τριτοβάθμιου ιδρύματος .

**ΠΙΝΑΚΑΣ STUD\_COUR\_TEACH**

Student	Course	Mark	Teacher
ΔΟΥΜΑ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	10	ΣΚΟΥΡΛΑΣ
ΜΑΥΡΟΥΔΑΚΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	10	ΜΙΑΟΥΛΗΣ
ΠΑΠΟΥΤΣΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ	8	ΜΙΑΟΥΛΗΣ
ΔΟΥΜΑ	PROLOG	9	ΞΑΝΘΑΚΗΣ
ΜΑΥΡΟΥΔΑΚΗΣ	PROLOG	8	ΚΑΤΣΙΚΑΣ

Για τον πίνακα αυτό μπορούμε να πούμε ότι βρίσκεται στην τρίτη κανονική μορφή και το κύριο κλειδί του είναι το σύνθετο κλειδί (Student, Course).



## Περιορισμός

Αν ένας διδάσκων διδάσκει ακριβώς ένα μάθημα θα μπορούσε το χαρακτηριστικό Teacher να καθορίζει το χαρακτηριστικό Course. Δηλαδή, στο παράδειγμά μας έχουμε μία (σχετικά ασυνήθιστη) περίπτωση όπου χαρακτηριστικό εκτός κλειδιού ορίζει τμήμα του σύνθετου κλειδιού ενός πίνακα της τρίτης κανονικής μορφής.

Η κανονική μορφή Boyce-Codd έχει δύο πίνακες :

**ΠΙΝΑΚΑΣ STUDENT**

Student	Teacher	Mark
ΔΟΥΜΑ	ΣΚΟΥΡΛΑΣ	10
ΜΑΥΡΟΥΔΑΚΗΣ	ΜΙΑΟΥΛΗΣ	10
ΠΑΠΟΥΤΣΗΣ	ΜΙΑΟΥΛΗΣ	8
ΔΟΥΜΑ	ΞΑΝΘΑΚΗΣ	9
ΜΑΥΡΟΥΔΑΚΗΣ	ΚΑΤΣΙΚΑΣ	8

κύριο κλειδί = (Student, Teacher)

**ΠΙΝΑΚΑΣ COUR\_TEACH**

Teacher	Course
ΣΚΟΥΡΛΑΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
ΜΙΑΟΥΛΗΣ	ΒΑΣΕΙΣ ΔΕΔΟΜΕΝΩΝ
ΞΑΝΘΑΚΗΣ	PROLOG
ΚΑΤΣΙΚΑΣ	PROLOG

κύριο κλειδί = Teacher

Αν μία σχέση είναι στη κανονική μορφή (BCNF) τότε είναι και στην τρίτη κανονική μορφή.

### Βάση προσωπικού

emp (πίνακας υπαλλήλου)

Empno	Ename	Job	Sal	Comm	Deptno
10	CODD	ANALYST	1200	150	10
20	MARTIN	SALESMAN	2000		20
30	DATE	ANALYST	1800	200	10

Emp\_proj (πίνακας υπαλλήλου)

Empno	Projno	ptime
10	100	75
10	200	25
20	100	30
20	200	70
30	100	100

dept (πίνακας τμημάτων)

Deptno	Dname
10	ACCOUNTING
20	SALES
30	PROJECTS

proj (πίνακας έργων)

Projno	Pdescr	Loc
100	PAYROLL	ATHENS
200	PERSONNEL	PARIS

Να προσθέσετε το χαρακτηριστικό job\_code (= κωδικός θέσης) και να ανασχεδιάσετε τη βάση. Σχεδιάστε το Μοντέλο Οντοτήτων–Συσχετίσεων με συμβολισμό Elmasri-Navathe

## Cursors

Έστω οι πίνακες

```
CREATE TABLE scoEMP(EMPNO NUMBER(4) NOT NULL,  
    ENAME VARCHAR2(10), JOB VARCHAR2(9),  
    MGR NUMBER(4), HIREDATE DATE,  
    SAL NUMBER(7,2), COMM NUMBER(7,2),  
    DEPTNO NUMBER(2),  
    PRIMARY KEY(EMPNO));  
CREATE TABLE Emp_sal(msg VARCHAR2(20),  
    sal NUMBER(6), ename CHAR(10))
```

Γράψτε πρόγραμμα που χρησιμοποιεί Cursor. Το πρόγραμμα βρίσκει όλους τους υπαλλήλους με μηνιαία αμοιβή μεγαλύτερη από 2000 ευρώ, μεταξύ 1000 και 2000 ευρώ και κάτω από 1000 ευρώ. Για κάθε υπάλληλο ενημερώνει σχετικά τον πίνακα Emp\_sal με την ένδειξη >2000, >=1000 AND <=2000, >2000 ανάλογα με την περίπτωση.

### **Εντολή (Statement ) IF-THEN-ELSIF**

IF condition THEN

statement; ... statement;

[ELSIF condition THEN

statement; ... statement;]

...

[ELSIF condition THEN

statement; ... statement;]

[ELSE

statement; ... statement;]

END IF;

## IF Syntax

### mySQL

```
IF search_condition THEN statement_list
    [ELSEIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF;
```

### Oracle

```
IF search_condition THEN statement_list
    [ELSIF search_condition THEN statement_list] ...
    [ELSE statement_list]
END IF;
```



# Ερωτήσεις