



Ανοικτά Ακαδημαϊκά Μαθήματα στο Πανεπιστήμιο Δυτικής Αττικής

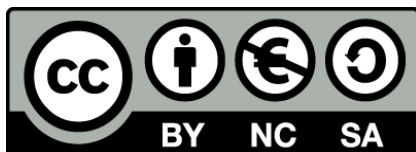


# Βάσεις Δεδομένων I (ICE1-4001)

## Ενότητα 1: «Προσανατολισμού» (orientation)

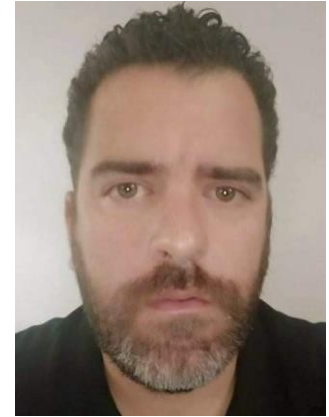
Π.Ανδρίτσος

Ρ.Γαροφαλάκη, Θ.Παυλίδης, Α.Τσολακίδης



Το περιεχόμενο του μαθήματος  
διατίθεται με άδεια Creative  
Commons εκτός και αν αναφέρεται  
διαφορετικά

# Εναρκτήρια συνάντηση



Στην πρώτη συνάντηση γίνεται παρουσίαση του μαθήματος και μία σύντομη και περιεκτική επισκόπηση κάποιων βασικών εννοιών των Βάσεων Δεδομένων.

Η διεκπεραίωση των θεμάτων γίνεται κυρίως με χρήση παραδειγμάτων.



# Τι είναι οι βάσεις δεδομένων

γαλλικά: Base de données,  
αγγλικά: database, γερμανικά:  
Datenbank

Μία βάση δεδομένων είναι ένα είδος ηλεκτρονικής αρχειοθέτησης δεδομένων (στοιχείων, data) ενός οργανισμού ή μιας επιχείρησης ή ακόμη και ενός φυσικού προσώπου, π.χ., ενός μεμονωμένου επαγγελματία. Μαζί με τα στοιχεία υπάρχει το σύνολο των εφαρμογών που επιτρέπουν στους χρήστες της βάσης να καταχωρήσουν και να ανακτήσουν τα στοιχεία αυτά.

# Βάση Δεδομένων (database)

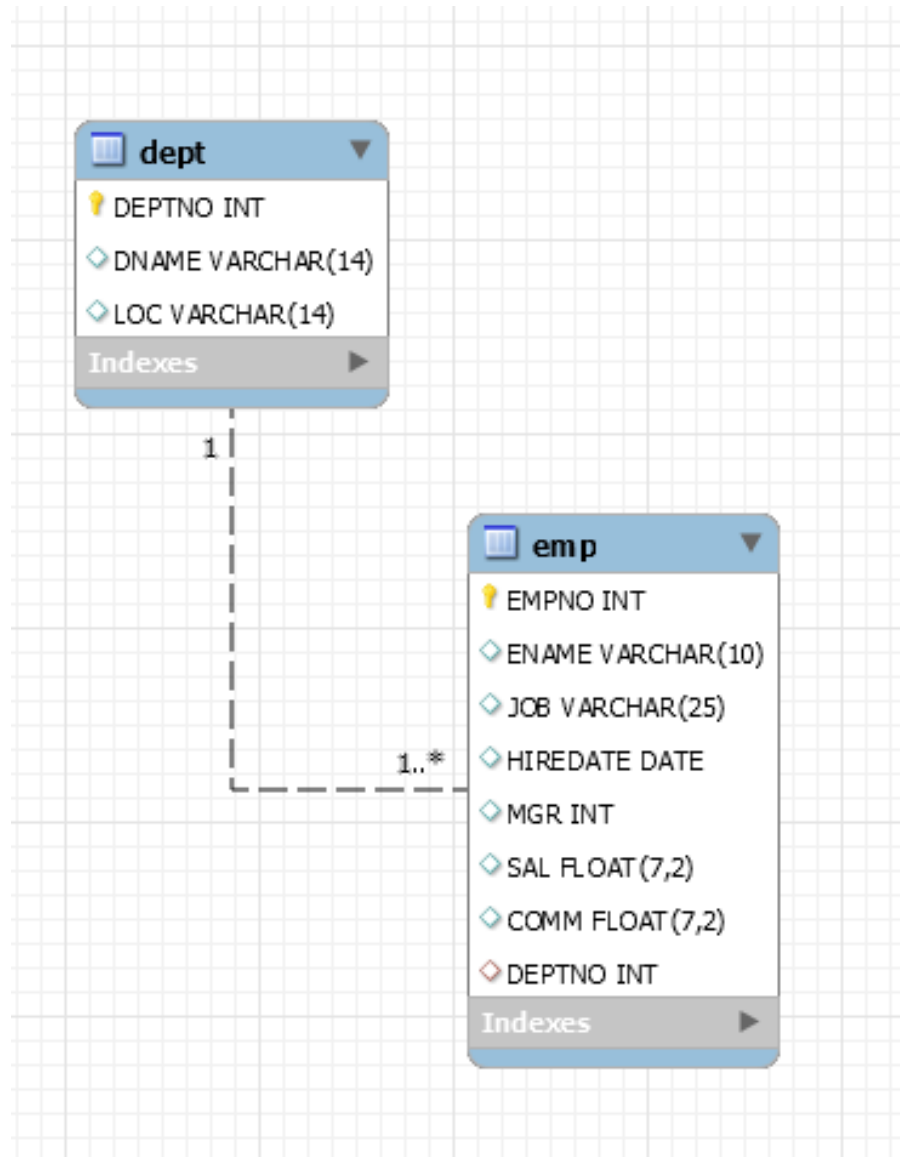
- Σε μία εκπαιδευτική βάση παράδειγμα Δεδομένων είναι τα στοιχεία φοιτητών και καθηγητών, οι βαθμολογίες, οι δηλώσεις των μαθημάτων κατά την εγγραφή του σπουδαστή κ.λπ. Παράδειγμα εφαρμογής είναι τα προγράμματα (application software) που «αναλαμβάνουν» τη διαχείριση του φοιτητολογίου, δηλαδή των εγγραφών, των βαθμολογιών των σπουδαστών κ.λπ.

# Σύστημα Βάσης Δεδομένων

Ένα Σύστημα Βάσης Δεδομένων είναι ένα σύστημα καταχώρησης, ενημέρωσης και ανάκτησης δεδομένων βασιζόμενο σε υπολογιστή και αποτελείται από συνιστώσες:

- Δεδομένα (Data)
- Υλικό (Hardware)
- Λογισμικό (Software) , με κυριότερο στοιχείο του το **Σύστημα Διαχείρισης Βάσεων Δεδομένων** (π.χ., Oracle, mySQL), και **εφαρμογές λογισμικού** για τους τελικούς χρήστες (end-users).
- Χρήστες (end-users)

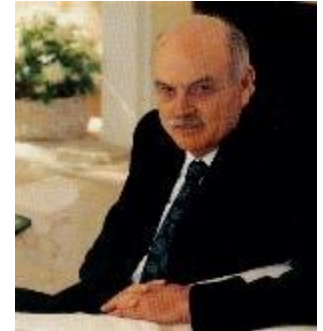
# Παράδειγμα εννοιολογικού μοντέλου σε MySQL Workbench (Unified Modeling Language – UML)



# **Σχεσιακές βάσεις δεδομένων – Tedd Codd**

Σχεσιακό μοντέλο δεδομένων, σύμφωνα με την Ορολογία του Codd

	Attribute			
Tuple				
< ----- Relation ----- >				



Σχεσιακό μοντέλο δεδομένων, σύμφωνα με την Ορολογία των Προϊόντων Διαχείρισης Βάσης Δεδομένων

	Column			
Row				
< ----- Table ----- >				

```
mysql> SELECT * FROM DEPT;
+-----+-----+-----+
| DEPTNO | DNAME      | LOC    |
+-----+-----+-----+
| 10     | ACCOUNTING | NEW YORK |
| 20     | RESEARCH   | DALLAS  |
| 30     | SALES      | CHICAGO |
| 40     | OPERATIONS | BOSTON  |
+-----+-----+-----+
4 rows in set (0.04 sec)
```



# Περιγραφή Μαθήματος

- Το μάθημα αποτελεί εισαγωγή στην τεχνολογία και τις έννοιες των Βάσεων Δεδομένων (ΒΔ), των Συστημάτων Βάσεων Δεδομένων (ΣΒΔ) και στην ανάπτυξη εφαρμογών με χρήση Συστημάτων Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ).
- Η ύλη του στοχεύει στην εισαγωγή στις απαραίτητες βασικές έννοιες των ΒΔ, των ΣΒΔ και των ΣΔΒΔ ώστε οι φοιτητές να κατανοήσουν την τεχνολογία των ΒΔ, των ΣΒΔ και των Προϊόντων Διαχείρισης Βάσεων Δεδομένων (ΠΔΒΔ). Έμφαση δίδεται στην παρουσίαση των εννοιών της σχεδίασης ΒΔ (μοντελοποίηση-κανονικοποίηση) και στην υλοποίηση με χρήση ΠΔΒΔ που υποστηρίζουν τη γλώσσα SQL, έτσι ώστε οι φοιτητές να έχουν μία συνολική αντίληψη των διαδικασιών και μεθοδολογιών σχεδιασμού και υλοποίησης ΣΒΔ.

# Μαθησιακά Αποτελέσματα

Κύριος στόχος του μαθήματος είναι να εφοδιάσει τους φοιτητές με τις απαραίτητες γνώσεις έτσι ώστε να είναι ικανοί να σχεδιάσουν ΒΔ και ΣΒΔ και να υλοποιήσουν ΒΔ με χρήση γλώσσας SQL.

Με την επιτυχή ολοκλήρωση του μαθήματος οι φοιτητές:

- Θα έχουν κατανοήσει τα βασικά εργαλεία της τεχνολογίας ΒΔ και γνωστών ΠΔΒΔ,
- Θα είναι σε θέση να αναλύσουν επιχειρηματικούς κανόνες-περιορισμούς για να σχεδιάσουν ΒΔ,
- Θα είναι σε θέση να εφαρμόσουν τις βασικές τεχνικές σχεδίασης και υλοποίησης ΒΔ,
- Θα είναι σε θέση να εφαρμόσουν τις βασικές τεχνικές χρήσης γλώσσας SQL για την υλοποίηση ΣΒΔ,
- Θα έχουν κατανοήσει βασικά θέματα συναλλαγών (transactions), διαχείρισης βάσεων (database administration) και διαχείρισης όψεων (views)

# Περίγραμμα ύλης μαθήματος

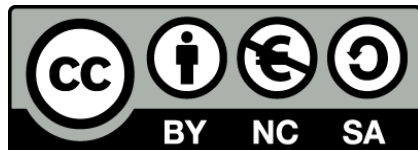
1. Βάσεις δεδομένων, Συστήματα Βάσεων Δεδομένων, Συστήματα Διαχείρισης Βάσεων Δεδομένων, Αρχιτεκτονική Συστημάτων Διαχείρισης Βάσεων Δεδομένων.
2. Δομές Δεδομένων για Βάσεις Δεδομένων. Κλασσικά μοντέλα βάσεων δεδομένων (Ιεραρχικό, Δικτυωτό). Σχεσιακό μοντέλο δεδομένων. Σχεσιακή άλγεβρα, Σχεσιακός λογισμός και QBE.
3. Μοντελοποίηση. Ενοσιολογικά μοντέλα. Μοντέλο Οντοτήτων-Συσχετίσεων.
4. Λογικός σχεδιασμός σχεσιακών βάσεων δεδομένων. Κανονικοποίηση. Συναρτησιακές εξαρτήσεις και κανονικοποίηση.
5. Γλώσσες για βάσεις δεδομένων. Γλώσσα SQL. Πρότυπο SQL3.
6. Όψεις (views). Ενημερωσιμότητα όψεων.
7. Θέματα διαχείρισης και λειτουργίας βάσεων δεδομένων. Ακεραιότητα, βελτιστοποίηση, αναδιοργάνωση, ασφάλεια, λειτουργικότητα, κ.λπ.
8. Ο Ρόλος και τα καθήκοντα του Διαχειριστή Βάσεων Δεδομένων.

# Ενδεικτική Βιβλιογραφία

1. Jeffrey Ullman, Jennifer Widom, Βασικές αρχές για τα συστήματα βάσεων δεδομένων, έκδοση 2η, 2008, ISBN: 978-960-461-183-6, εκδόσεις Κλειδάριθμος
2. Elmasri Ramez, Navathe Shamkant, Θεμελιώδεις αρχές συστημάτων βάσεων δεδομένων, έκδοση 7η, αναθεωρημένη, 2016, εκδόσεις Δίαυλος
3. Ramakrishnan Raghu, Gehrke Joahannes, Συστήματα διαχείρισης βάσεων δεδομένων, έκδοση 3η, 2012, ISBN: 978-960-418-411-8, εκδόσεις Τζιόλα
4. Silberschatz A., Korth H.F., Sudarshan S., Συστήματα βάσεων δεδομένων – Η πλήρης θεωρία των βάσεων δεδομένων, έκδοση 6η, Εκδόσεις Γκιούρδα
5. Garcia-Molina, Ullman, Widom, Συστήματα βάσεων δεδομένων, έκδοση 1η, 2012, ISBN: 978-960-524-309-8, Πανεπιστημιακές Εκδόσεις Κρήτης
6. Date A.J., An introduction to database systems, vol.1, Addison-Wesley
7. Connolly T., Begg C., Database solutions. A step-by-step guide to building databases, Addison-Wesley
8. Χρήστος Σκουρλάς, Σχεσιακές βάσεις δεδομένων, έκδοση 1η, 2000, ISBN: 960-8105-14-5, εκδόσεις Νέων Τεχνολογιών
9. Βασίλειος Ταμπακάς, Εισαγωγή στις βάσεις δεδομένων, έκδοση 1η, 2017, ISBN: 978-960-9427-66-1, εκδόσεις Γκότση
10. Εμμανουήλ Γιαννακουδάκης, Βάσεις Δεδομένων, Έκδοση Α', 2014, ISBN: 978-960-359-114-6, εκδόσεις Μπένου
11. Ιωάννης Μανωλόπουλος, Απόστολος Παπαδόπουλος, Συστήματα Βάσεων Δεδομένων, έκδοση 1η, 2006, ISBN: 960-8105-87-0, εκδόσεις Νέων Τεχνολογιών

- Τεχνικές αναφορές από ερευνητικά έργα, κεφάλαια διπλωματικών εργασιών, παραπομπές σε ανασκοπήσεις (review papers) και σε άρθρα σε ερευνητικά θέματα αιχμής για τις βάσεις δεδομένων και τις εφαρμογές τους.

# Τέλος Ενότητας



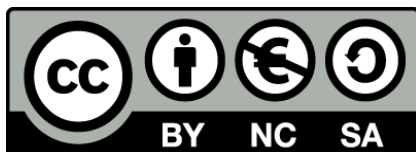


# Βάσεις Δεδομένων I (ICE1-4001)

## Ενότητα 2: Μοντέλο Οντοτήτων - Συσχετίσεων

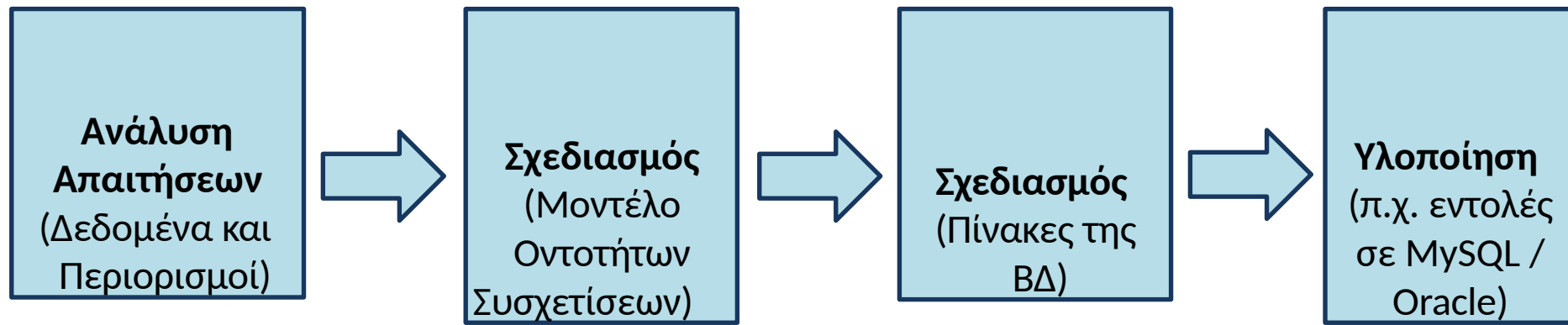
Π.Ανδρίτσος

Ρ.Γαροφαλάκη, Θ.Παυλίδης, Α.Τσολακίδης



Το περιεχόμενο του μαθήματος  
διατίθεται με άδεια Creative  
Commons εκτός και αν αναφέρεται  
διαφορετικά

# Μεθοδολογία



# Η μοντελοποίηση

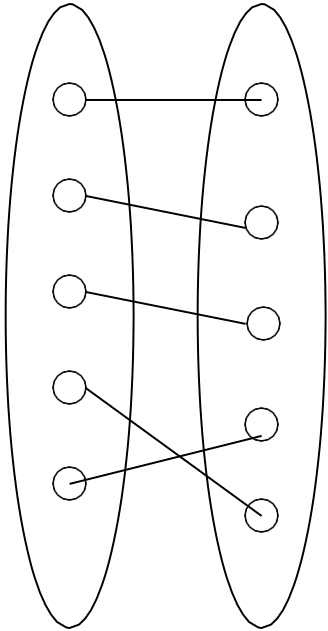
- Όταν θέλουμε να μοντελοποιήσουμε ένα σύστημα βάσης δεδομένων σχεδιάζουμε **ένα** ειδικό μοντέλο για όλες τις εφαρμογές που μας ενδιαφέρουν, το **Μοντέλο Οντοτήτων Συσχετίσεων (ΜΟΣ)**. Το μοντέλο αναπαριστά όλες τις οντότητες (entities) και τις μεταξύ τους συσχετίσεις (relationships).



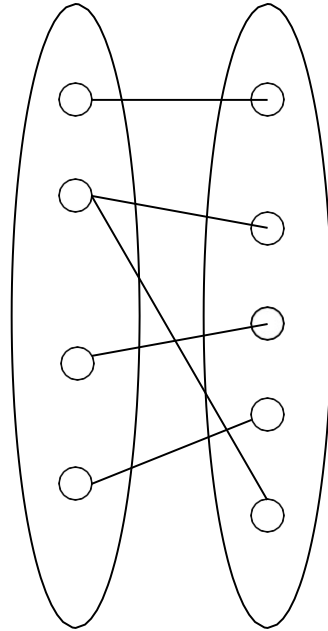
# Βαθμός Συσχέτισης

- 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).

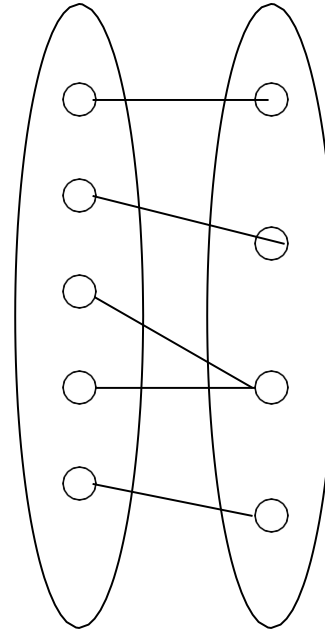
# Δυαδικές (binary) Συσχετίσεις στο μοντέλο Οντοτήτων-Συσχετίσεων



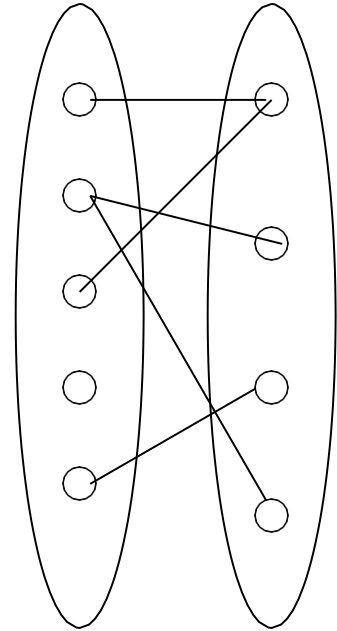
1-προς-1  
Περιφέρεια  
& Έδρα



1-προς-Πολλά  
Πελάτης &  
Παραγγελί  
α

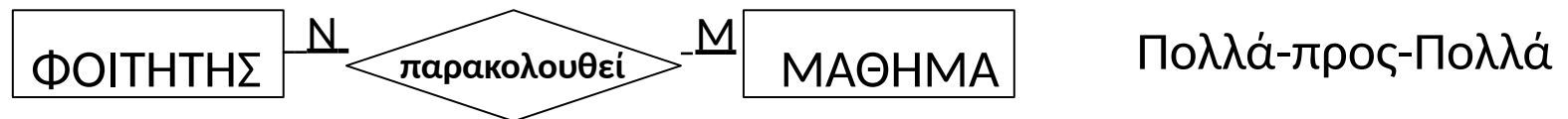
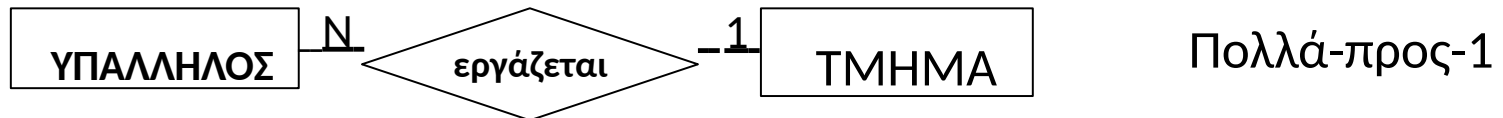
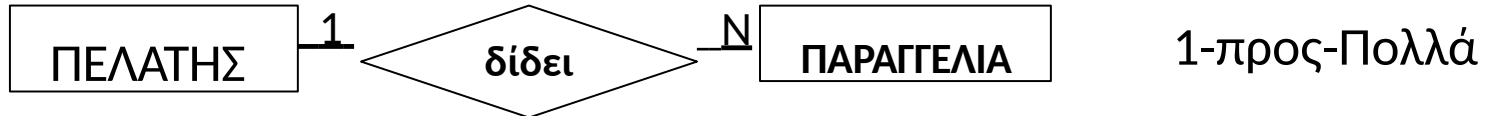
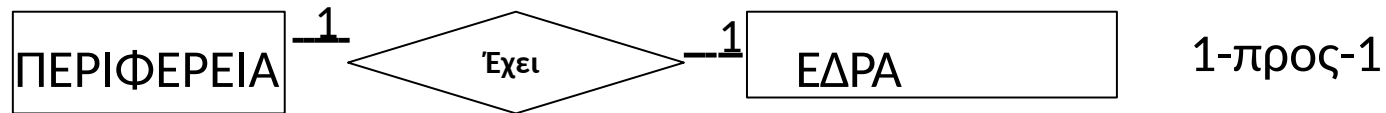


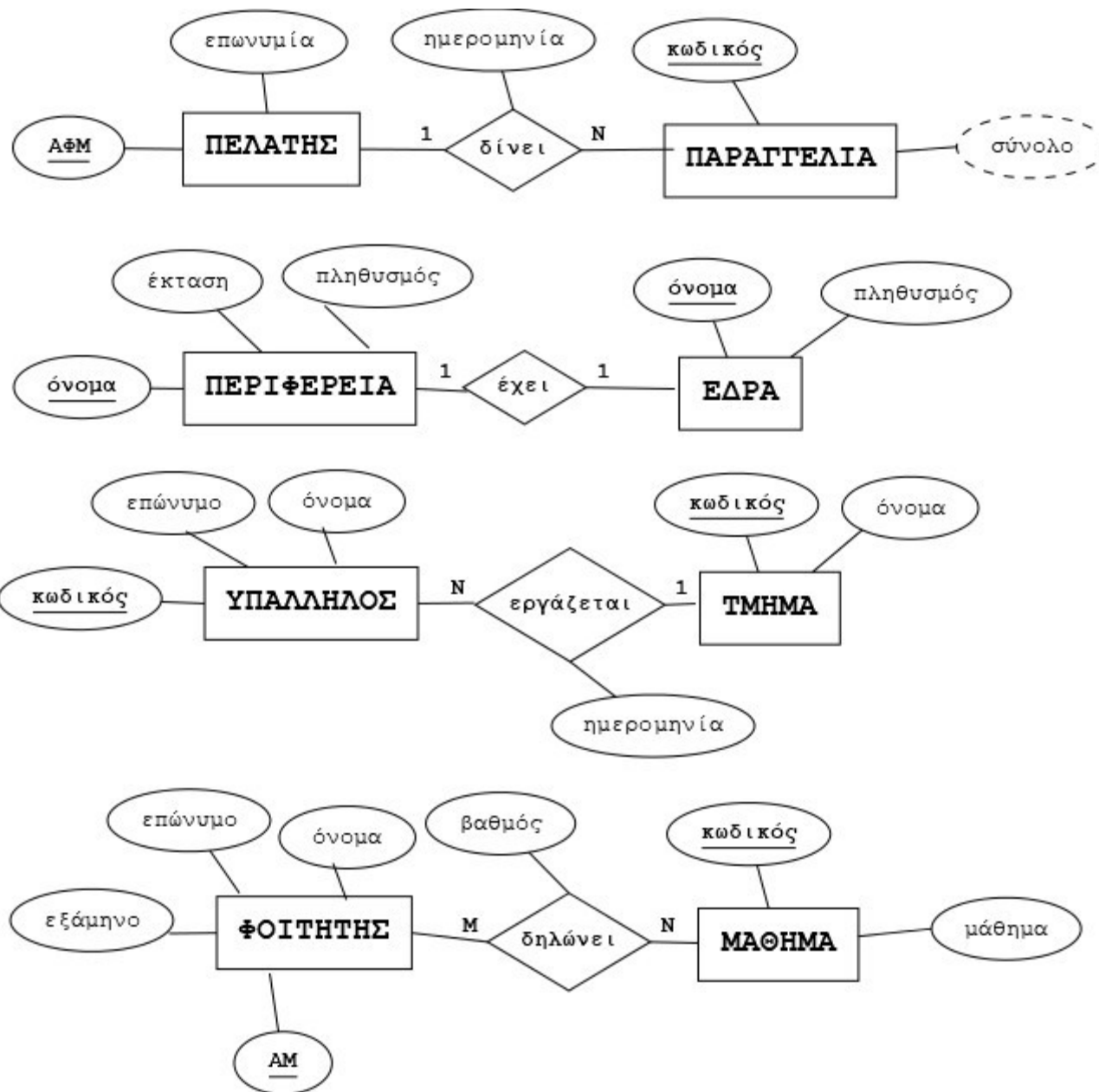
Πολλά-προς-1  
Υπάλληλος &  
Τμήμα



Πολλά-προς-  
Πολλά  
Φοιτητής &  
Μάθημα

# Παραδείγματα Δυαδικών Συσχετίσεων





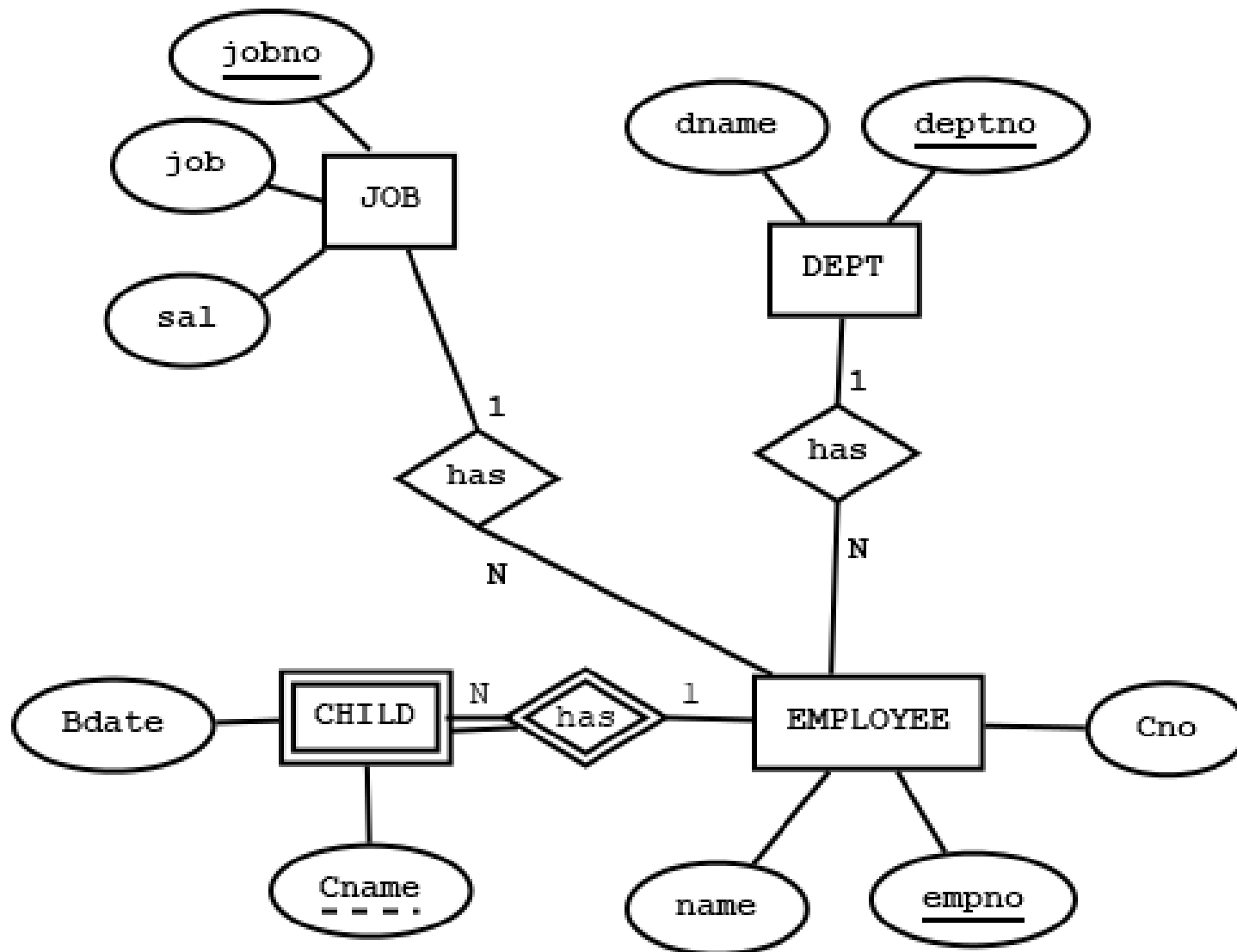
# Σχεδίαση Βάσης δεδομένων Διεύθυνσης Προσωπικού

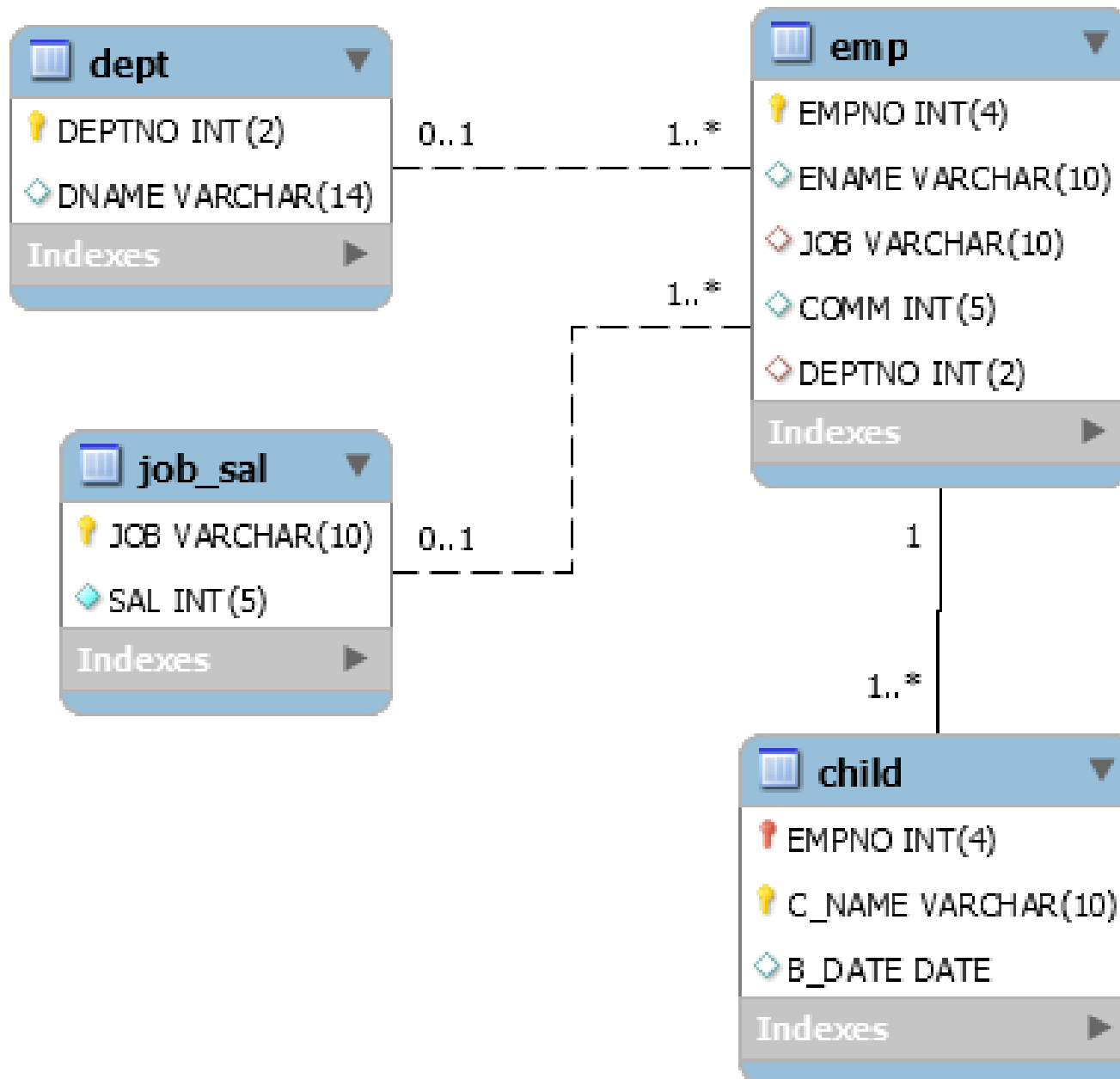
Οι στήλες των πινάκων είναι οι εξής:

Empno=Κωδικός υπαλλήλου, Surname=επώνυμο,  
Name=όνομα, Job=θέση, Deptno=κωδικός  
τμήματος, Dname=τμήμα Sal=μισθός,  
C\_No=κωδικός παιδιού 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 μπορεί να έχει ή να μην έχει παιδιά.





# Υλοποίηση βάσης δεδομένων Διεύθυνσης Προσωπικού

Πίνακας υπαλλήλου

*emp*

ENAME	ENO	JNO	COMM	DNO
ΣΠΥΡΟΥ	10	100	150	10
ΧΡΗΣΤΟΥ	20	100	250	10
ΧΡΗΣΤΟΥ	30	200		20
ΝΙΚΟΥ	40	100		10

Πίνακας τμήματος

*dept*

DNO	DEPT
10	ΛΟΓΙΣΤΗΡΙΟ
20	ΠΩΛΗΣΕΙΣ
30	ΠΡΟΣΩΠΙΚΟΥ

Πίνακας θέσης

*job*

JNO	JOB	SAL
100	ΑΝΑΛΥΤΗΣ	2500
200	ΠΩΛΗΤΗΣ	2000



# Υλοποίηση βάσης δεδομένων Διεύθυνσης Προσωπικού

DROP DATABASE exams;

CREATE DATABASE

exams;

CREATE TABLE dept(Dno INT(2) NOT NULL, Dept VARCHAR(40), PRIMARY  
KEY(Dno));

CREATE TABLE job(Jno INT(2) NOT NULL, job VARCHAR(40), sal FLOAT(7,2),  
PRIMARY KEY(Jno));

CREATE TABLE emp(Ename VARCHAR(30), Eno INT(4) NOT NULL, JNo INT(3),  
Comm FLOAT(7,2), Dno INT(2), PRIMARY KEY(Eno),  
FOREIGN KEY(JNo) REFERENCES job(JNo),  
FOREIGN KEY(Dno) REFERENCES dept(Dno));

INSERT INTO dept(Dno, Dept) VALUES  
(10, 'ΛΟΓΙΣΤΗΡΙΟ'), (20, 'ΠΩΛΗΣΕΙΣ'), (30,  
'ΤΙΡΟΣΩΠΙΚΟΥ');

INSERT INTO job(JNo, job, Sal) VALUES  
(100, 'ΑΝΑΛΥΤΗΣ', 2500), (200, 'ΠΩΛΗΤΗΣ',  
2000);

INSERT INTO emp(Ename, Eno, JNo, Comm, Dno) VALUES  
(('ΧΡΗΣΤΟΥ', 10, 100, 250,  
'ΧΡΗΣΤΟΥ', 10, 100, 10, NULL),  
20),  
(('ΝΙΚΟΥ', 40, 100, NULL, 10);

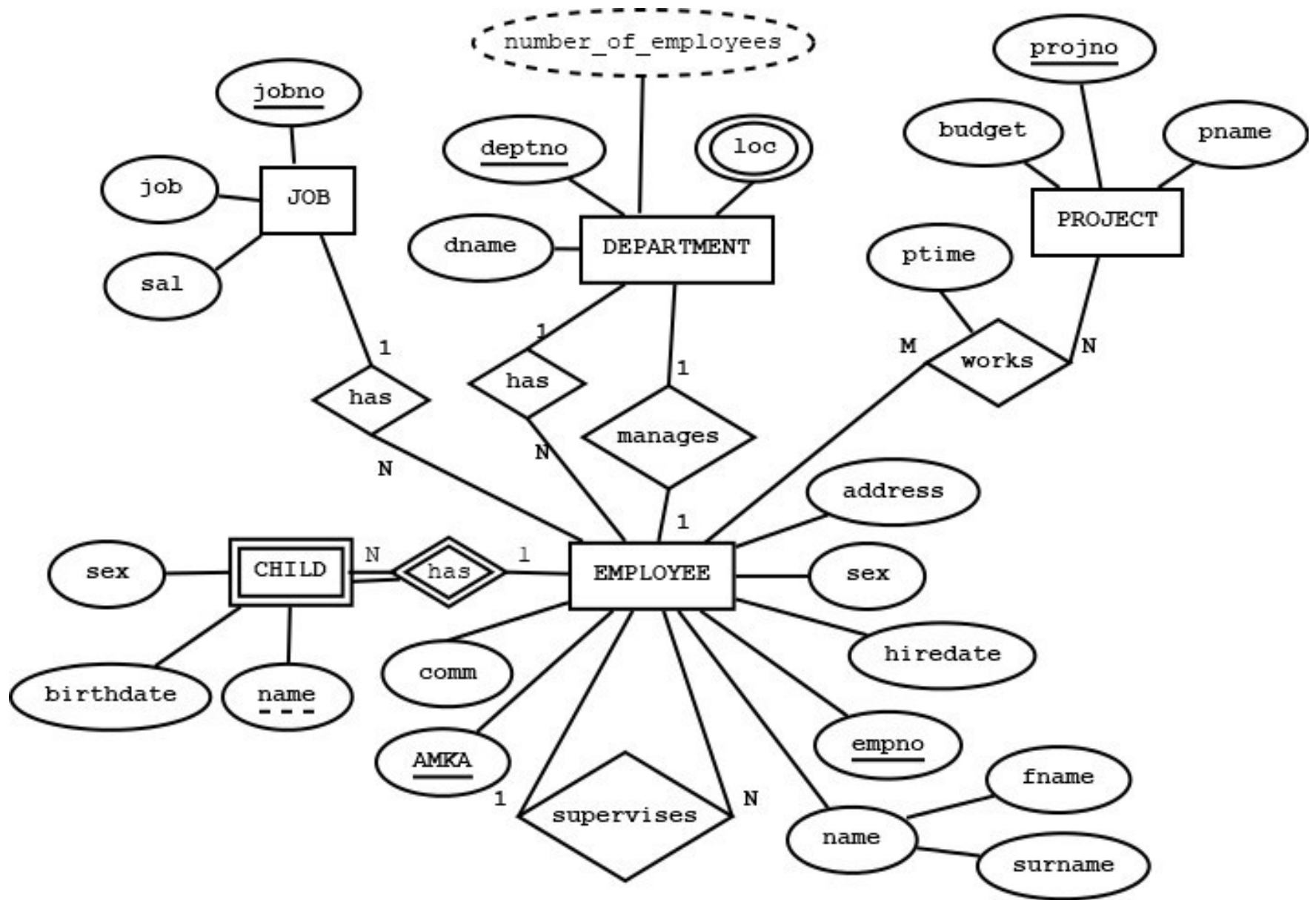
SELECT \* FROM DEPT;

SELECT \* FROM JOB;

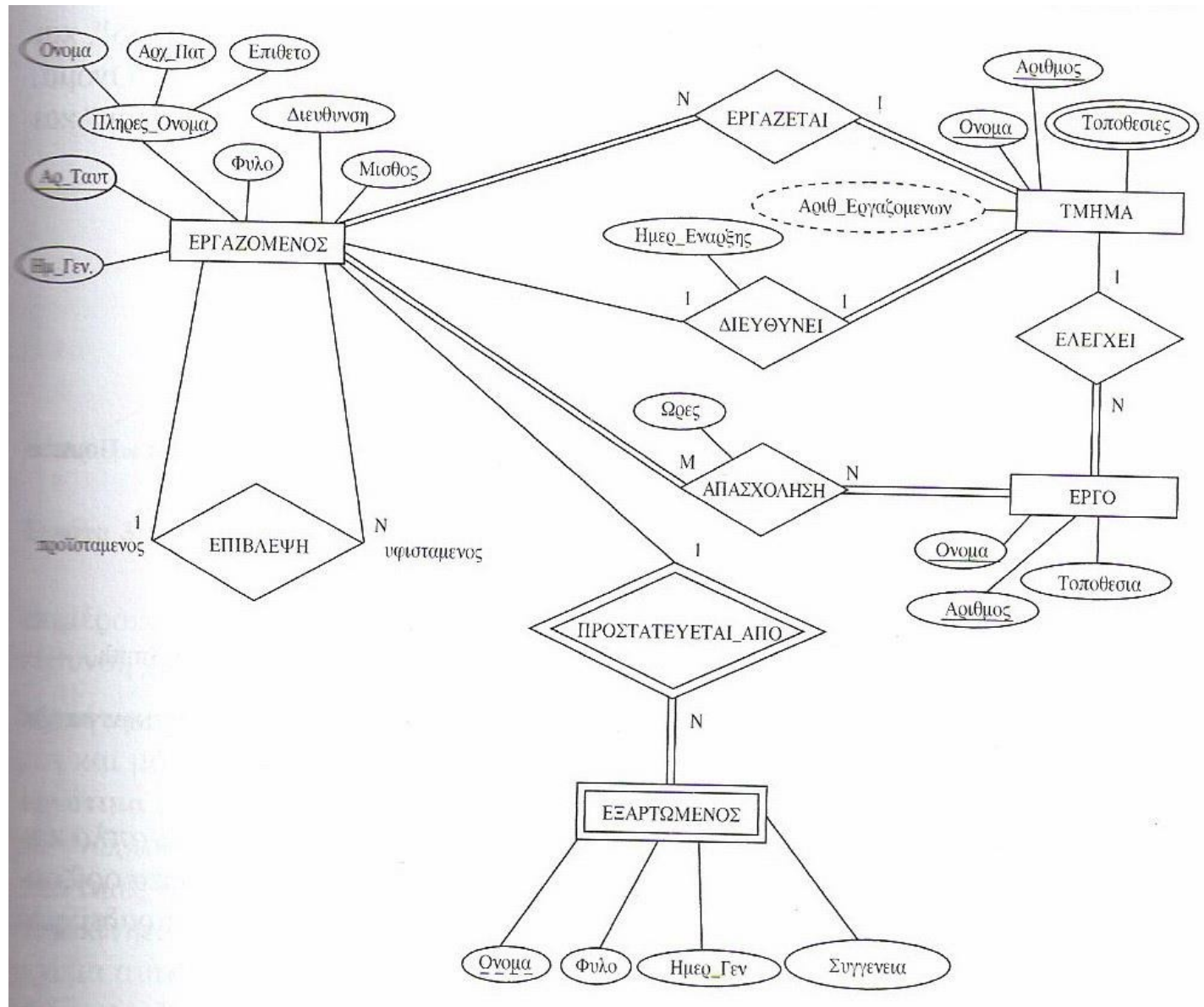
SELECT \* FROM EMP;

# Επέκταση της Βάσης δεδομένων Διεύθυνσης Προσωπικού με νέες Οντότητες

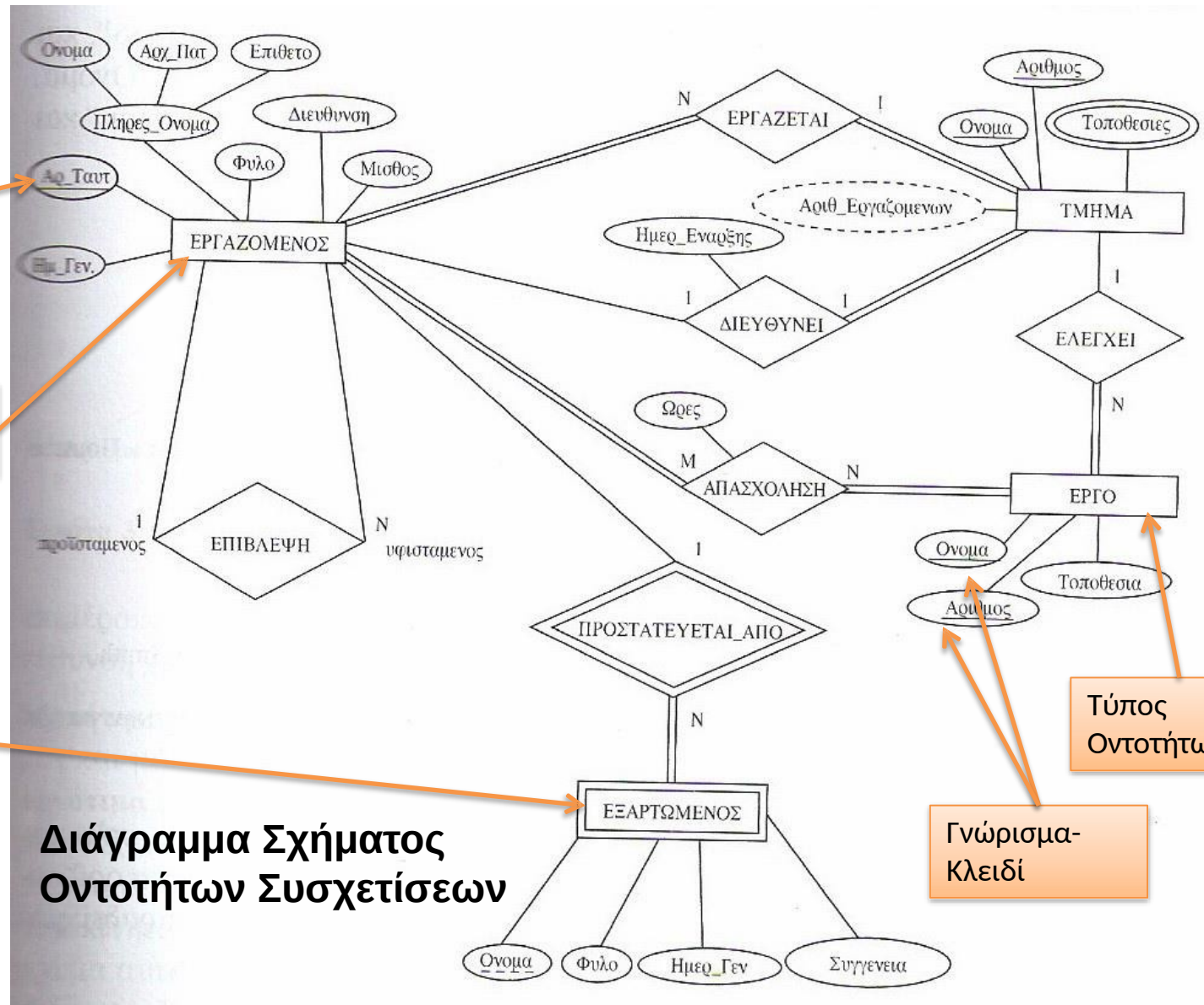
Department= τμήμα, deptno=κωδικός τμήματος,  
dname= όνομα, loc= έδρα, number\_of\_employees=  
αριθμός εργαζόμενων στο τμήμα,  
Employee= εργαζόμενος, empno= κωδικός,  
name=ονοματεπώνυμο, surname= επώνυμο, fname= όνομα,  
hiredate=ημερομηνία  
πρόσληψης, sex=φύλο, comm= προμήθεια, AFM= αριθμός  
φορολογικού μητρώου, Project= έργο, projno= κωδικός  
έργου, pname= όνομα, budget= προϋπολογισμός, ptime=  
ποσοστό χρόνου συμμετοχής εργαζόμενου σε έργο,  
Job= θέση, jobno= κωδικός  
θέσης, sal= μισθός,  
Child= τέκνο, name= όνομα, birthdate= ημερομηνία  
γέννησης,



# Διάγραμμα Σχήματος Οντοτήτων Συσχετίσεων



# Τύποι οντοτήτων



Γνώρισμα-Κλειδί

Τύπος Οντοτήτων

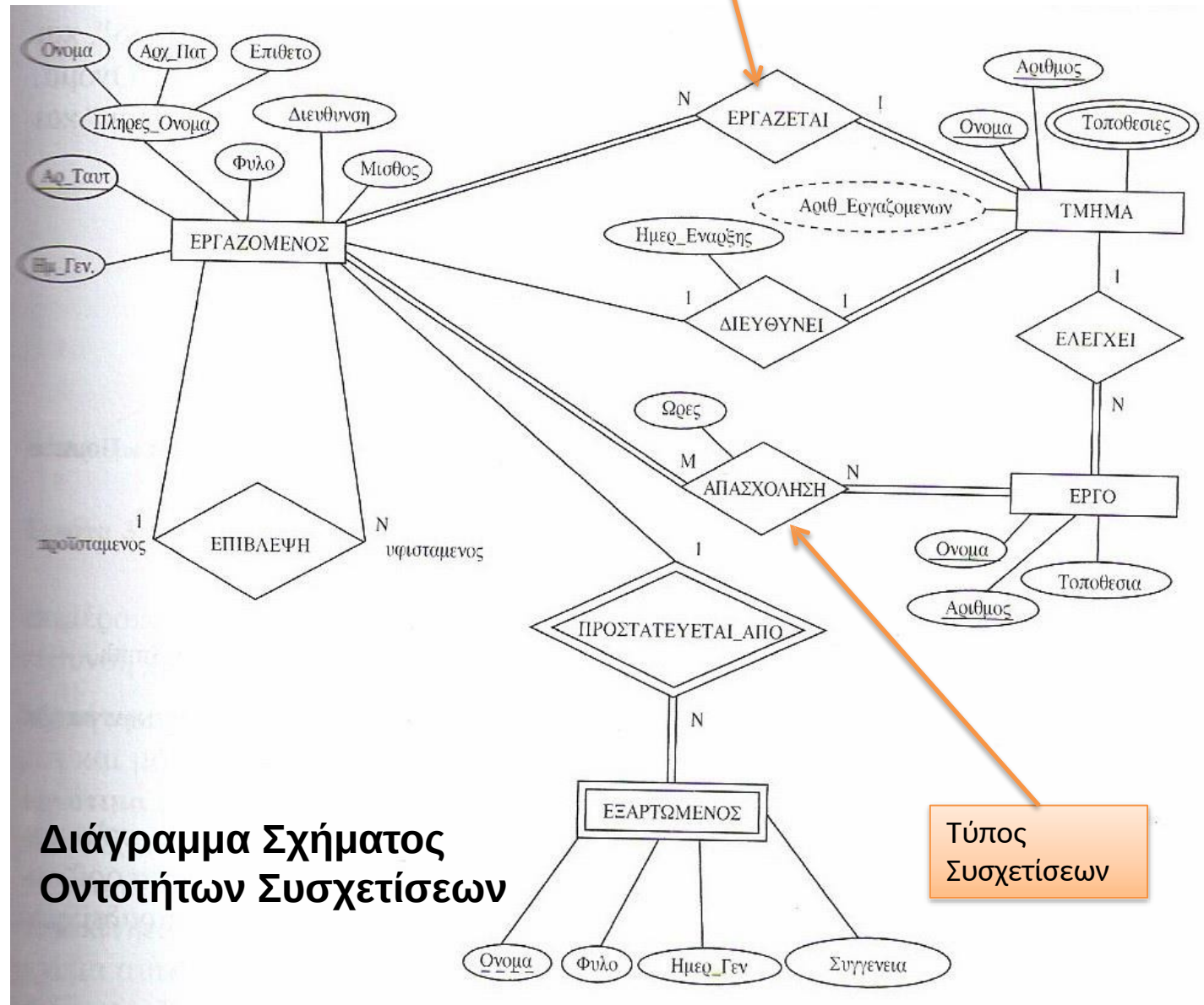
Μη Ισχυρός Τύπος Οντοτήτων

Διάγραμμα Σχήματος Οντοτήτων Συσχετίσεων

Τύπος Οντοτήτων

Γνώρισμα-Κλειδί

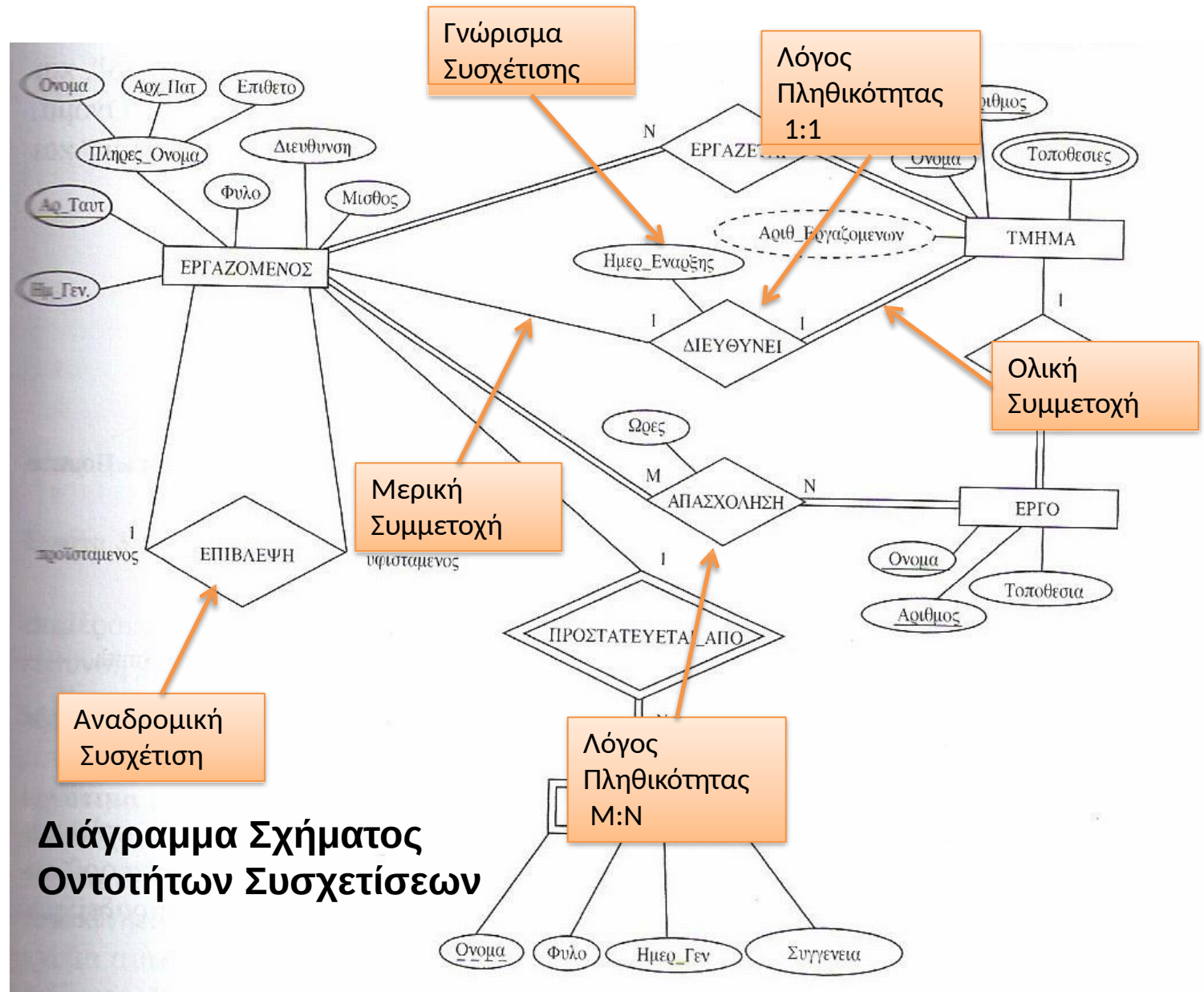
# Τύποι Συσχετίσεων



Διάγραμμα Σχήματος  
Οντοτήτων Συσχετίσεων



# Συσχετίσεις



Διάγραμμα Σχήματος  
Οντοτήτων Συσχετίσεων

# Σχεδιασμός Βάσης Δεδομένων Εκδόσεων Βιβλίων

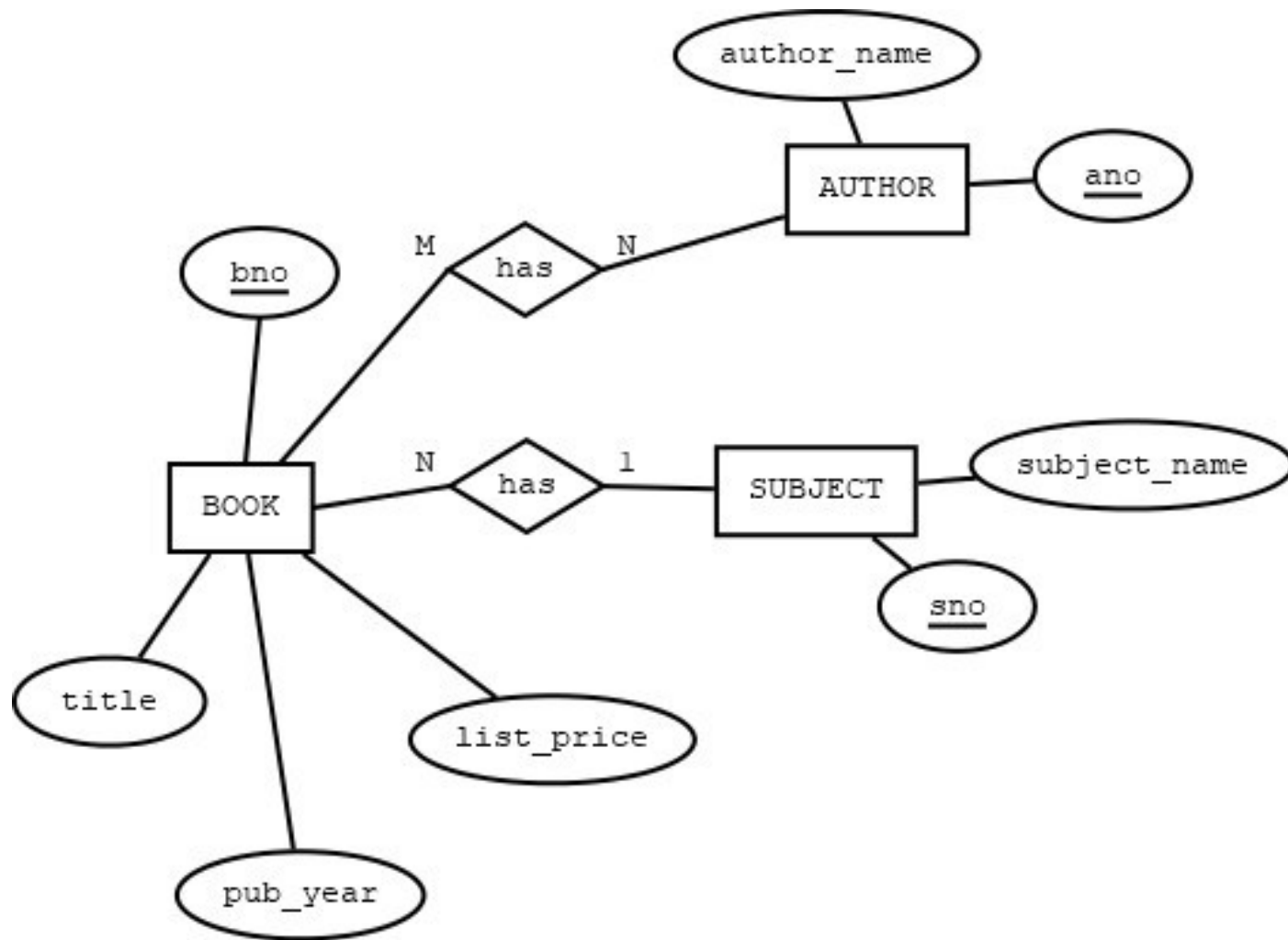
Ο εκδοτικός οίκος (Publisher) του Πανεπιστημίου σας αναθέτει τη σχεδίαση βάσης δεδομένων για τις εκδόσεις του. Μετά την ανάλυση των δεδομένων προκύπτουν οι παρακάτω περιορισμοί (constraints):

- Κάθε βιβλίο έχει ένα μοναδικό κωδικό (bno), τίτλο (title), έτος έκδοσης (pub\_year), τιμή καταλόγου (list\_price).
- Κάθε θέμα έχει έναν κωδικό (sno) και την ονομασία του θέματος (Subject).
- Κάθε βιβλίο μπορεί να έχει ένα θέμα (subject) και για ένα θέμα μπορεί να εκδίδονται πολλά βιβλία.
- Ένα βιβλίο μπορεί να έχει πολλούς συγγραφείς και ένας συγγραφέας έχει έναν κωδικό (auid) και το όνομά του.



# BOOKS (πίνακας στοιχείων βιβλίου)

Bno	Title	(Ano, Author)	Pub_ Year	List_ price	(Sno, Subject)
1	ΣΥΣΤΗΜΑΤΑ ΒΑΣΕΩΝ ΔΕΔΟΜΕΝΩΝ	(100, ULLMAN), (200, WIDOM)	2012	80	(1000, ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ)
2	Η ΜΕΘΟΔΟΣ PAGERANK	(300, LANGVILLE), (400, MEYER)	2010	35	(1000, ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ)
3	Η ΑΛΓΟΡΙΘΜΙΚΗ ΤΕΧΝΗ ΤΩΝ ΑΠΟΦΑΣΕΩΝ	(500, CHRISTIAN), (600, GRIFFITHS)	2018	18	(2000, ΕΠΙΣΤΗΜΟΝΙΚΗ ΕΚΛΑΪΚΕΥΣΗ)



**Τέλος Ενότητας**

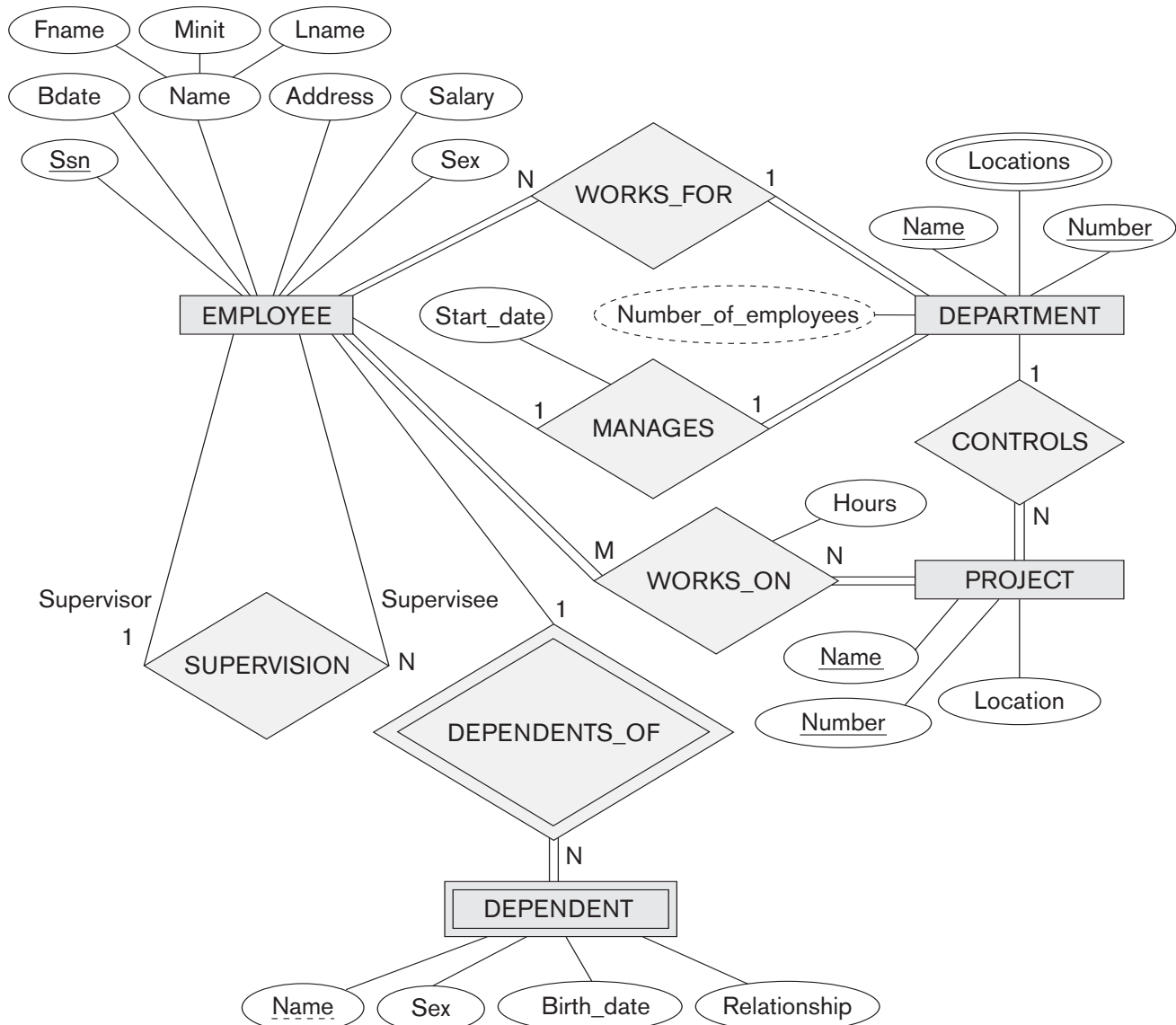
## 9.1 Relational Database Design Using ER-to-Relational Mapping

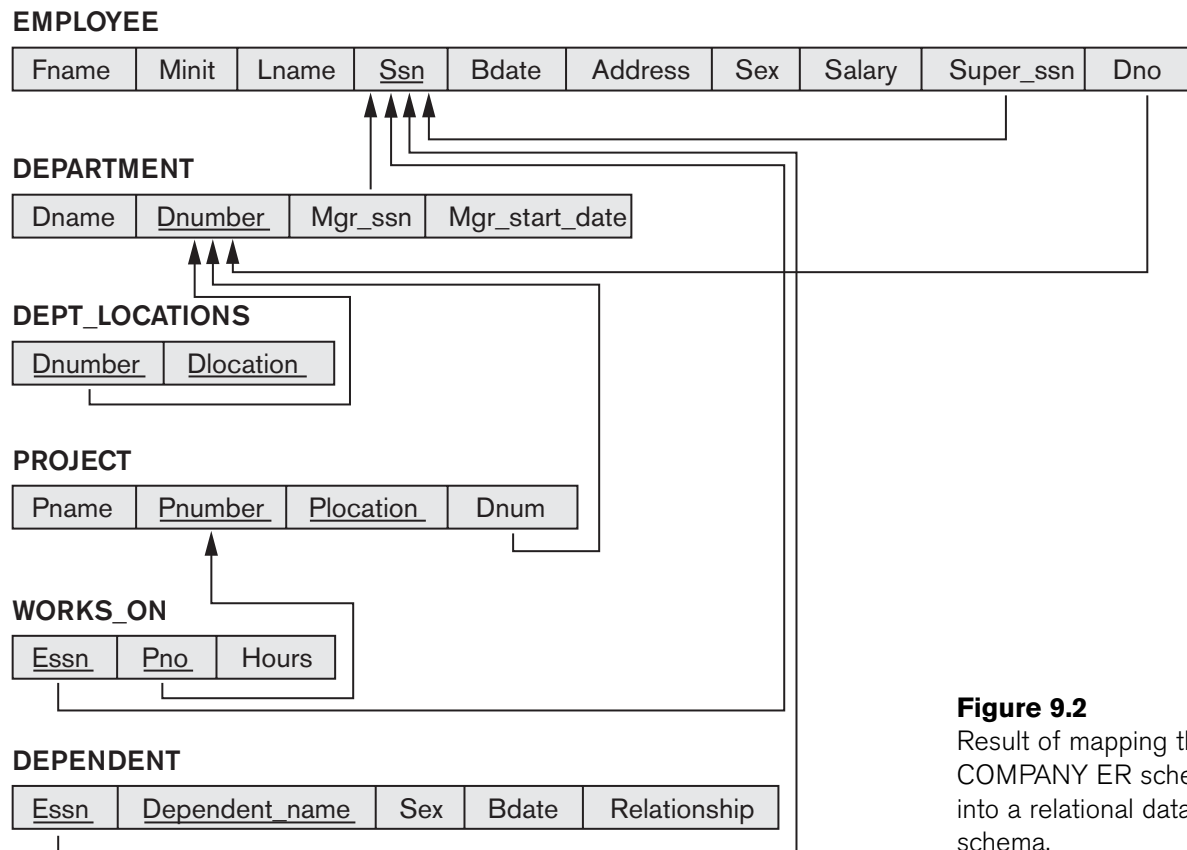
### 9.1.1 ER-to-Relational Mapping Algorithm

In this section we describe the steps of an algorithm for ER-to-relational mapping. We use the COMPANY database example to illustrate the mapping procedure. The COMPANY ER schema is shown again in Figure 9.1, and the corresponding COMPANY relational database schema is shown in Figure 9.2 to illustrate the map-

**Figure 9.1**

The ER conceptual schema diagram for the COMPANY database.





**Figure 9.2**  
Result of mapping the  
COMPANY ER schema  
into a relational database  
schema.

ping steps. We assume that the mapping will create tables with simple single-valued attributes. The relational model constraints defined in Chapter 3, which include primary keys, unique keys (if any), and referential integrity constraints on the relations, will also be specified in the mapping results.

**Step 1: Mapping of Regular Entity Types.** For each regular (strong) entity type  $E$  in the ER schema, create a relation  $R$  that includes all the simple attributes of  $E$ . Include only the simple component attributes of a composite attribute. Choose one of the key attributes of  $E$  as the primary key for  $R$ . If the chosen key of  $E$  is a composite, then the set of simple attributes that form it will together form the primary key of  $R$ .

If multiple keys were identified for  $E$  during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation  $R$ . Knowledge about keys is also kept for indexing purposes and other types of analyses.

In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.2 to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.1. The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the

The basic relational algebra does not have a NEST or COMPRESS operation that would produce a set of tuples of the form  $\{ \langle '1', 'Houston' \rangle, \langle '4', 'Stafford' \rangle, \langle '5', \{ 'Bellaire', 'Sugarland', 'Houston' \} \rangle \}$  from the DEPT\_LOCATIONS relation in Figure 3.6. This is a serious drawback of the basic normalized or *flat* version of the relational model. The object data model and object-relational systems (see Chapter 11) do allow multivalued attributes.

## 9.2 Mapping EER Model Constructs to Relations

Next, we discuss the mapping of EER model constructs to relations by extending the ER-to-relational mapping algorithm that was presented in Section 9.1.1.

### 9.2.1 Mapping of Specialization or Generalization

There are several options for mapping a number of subclasses that together form a specialization (or alternatively, that are generalized into a superclass), such as the {SECRETARY, TECHNICIAN, ENGINEER} subclasses of EMPLOYEE in Figure 8.4. We can add a further step to our ER-to-relational mapping algorithm from Section 9.1.1, which has seven steps, to handle the mapping of specialization. Step 8, which follows, gives the most common options; other mappings are also possible. We discuss the conditions under which each option should be used. We use  $\text{Attrs}(R)$  to denote *the attributes of relation R*, and  $\text{PK}(R)$  to denote *the primary key of R*. First we describe the mapping formally, then we illustrate it with examples.

**Step 8: Options for Mapping Specialization or Generalization.** Convert each specialization with  $m$  subclasses  $\{S_1, S_2, \dots, S_m\}$  and (generalized) superclass  $C$ , where the attributes of  $C$  are  $\{k, a_1, \dots, a_n\}$  and  $k$  is the (primary) key, into relation schemas using one of the following options:

- **Option 8A: Multiple relations—superclass and subclasses.** Create a relation  $L$  for  $C$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L) = k$ . Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 \leq i \leq m$ , with the attributes  $\text{Attrs}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$  and  $\text{PK}(L_i) = k$ . This option works for any specialization (total or partial, disjoint or overlapping).
- **Option 8B: Multiple relations—subclass relations only.** Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 \leq i \leq m$ , with the attributes  $\text{Attrs}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L_i) = k$ . This option only works for a specialization whose subclasses are *total* (every entity in the superclass must belong to (at least) one of the subclasses). Additionally, it is only recommended if the specialization has the *disjointness constraint* (see Section 8.3.1). If the specialization is *overlapping*, the same entity may be duplicated in several relations.
- **Option 8C: Single relation with one type attribute.** Create a single relation  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t\}$  and  $\text{PK}(L) = k$ . The attribute  $t$  is called a **type** (or

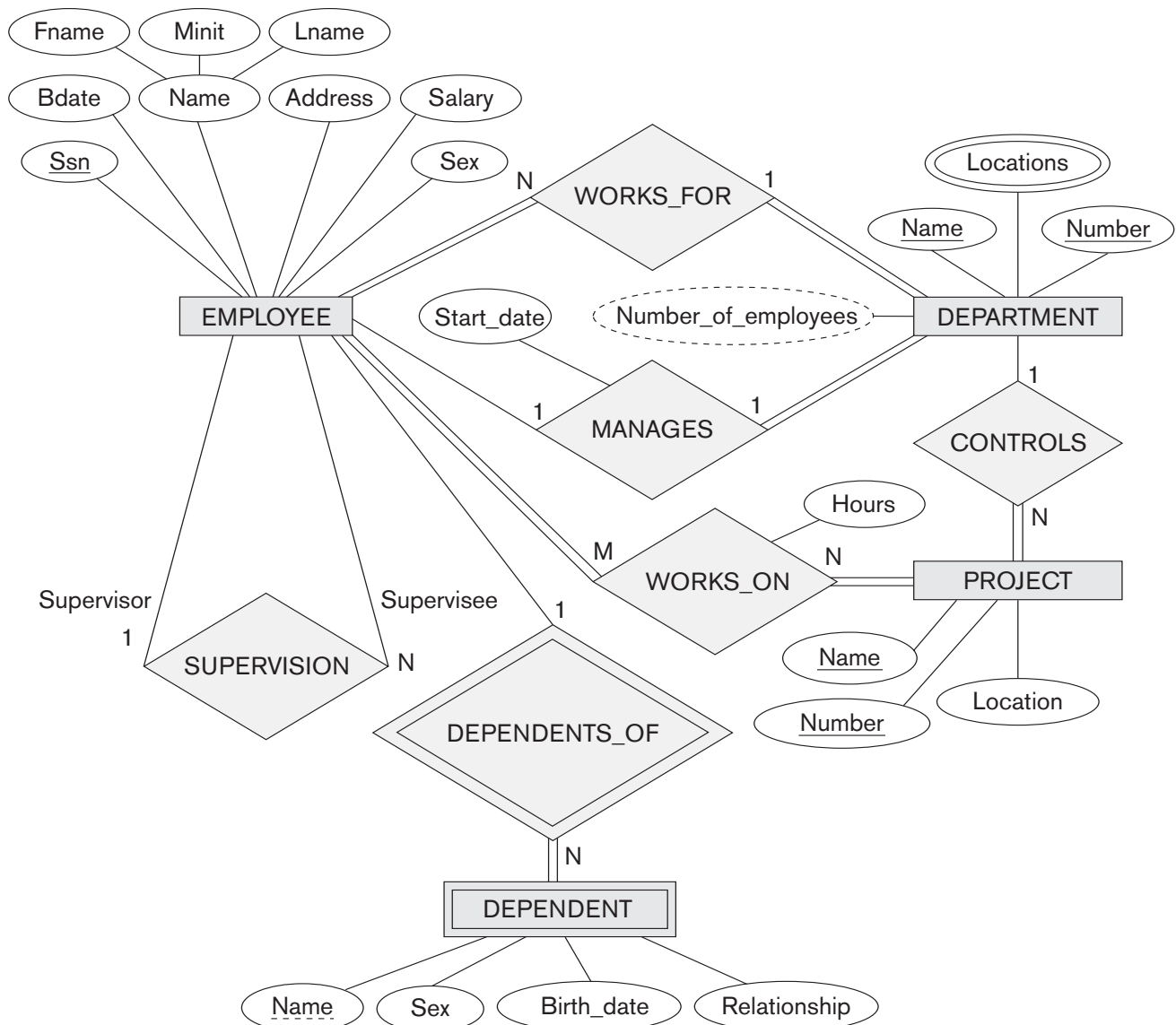
## 9.1 Relational Database Design Using ER-to-Relational Mapping

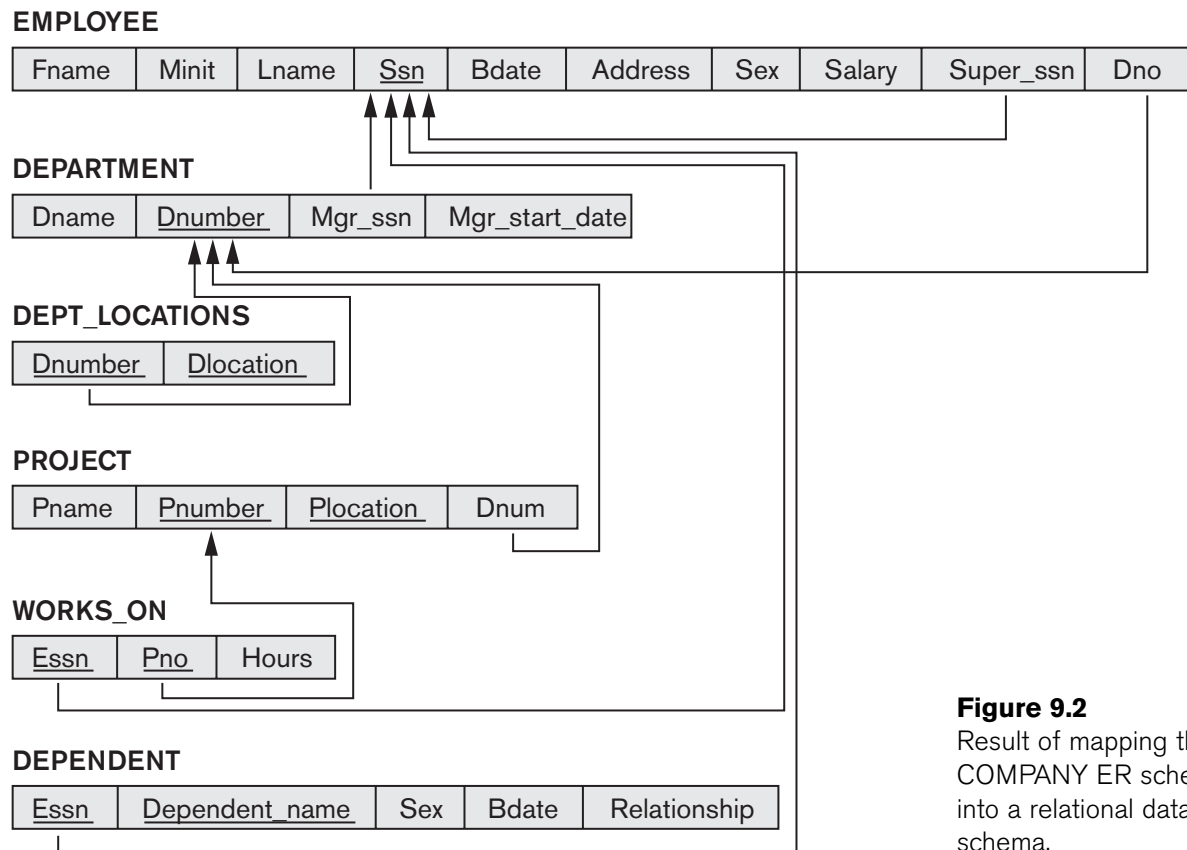
### 9.1.1 ER-to-Relational Mapping Algorithm

In this section we describe the steps of an algorithm for ER-to-relational mapping. We use the COMPANY database example to illustrate the mapping procedure. The COMPANY ER schema is shown again in Figure 9.1, and the corresponding COMPANY relational database schema is shown in Figure 9.2 to illustrate the map-

**Figure 9.1**

The ER conceptual schema diagram for the COMPANY database.





**Figure 9.2**  
Result of mapping the  
COMPANY ER schema  
into a relational database  
schema.

ping steps. We assume that the mapping will create tables with simple single-valued attributes. The relational model constraints defined in Chapter 3, which include primary keys, unique keys (if any), and referential integrity constraints on the relations, will also be specified in the mapping results.

**Step 1: Mapping of Regular Entity Types.** For each regular (strong) entity type  $E$  in the ER schema, create a relation  $R$  that includes all the simple attributes of  $E$ . Include only the simple component attributes of a composite attribute. Choose one of the key attributes of  $E$  as the primary key for  $R$ . If the chosen key of  $E$  is a composite, then the set of simple attributes that form it will together form the primary key of  $R$ .

If multiple keys were identified for  $E$  during the conceptual design, the information describing the attributes that form each additional key is kept in order to specify secondary (unique) keys of relation  $R$ . Knowledge about keys is also kept for indexing purposes and other types of analyses.

In our example, we create the relations EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.2 to correspond to the regular entity types EMPLOYEE, DEPARTMENT, and PROJECT in Figure 9.1. The foreign key and relationship attributes, if any, are not included yet; they will be added during subsequent steps. These include the



attributes Super\_ssn and Dno of EMPLOYEE, Mgr\_ssn and Mgr\_start\_date of DEPARTMENT, and Dnum of PROJECT. In our example, we choose Ssn, Dnumber, and Pnumber as primary keys for the relations EMPLOYEE, DEPARTMENT, and PROJECT, respectively. Knowledge that Dname of DEPARTMENT and Pname of PROJECT are secondary keys is kept for possible use later in the design.

The relations that are created from the mapping of entity types are sometimes called **entity relations** because each tuple represents an entity instance. The result after this mapping step is shown in Figure 9.3(a).

**Step 2: Mapping of Weak Entity Types.** For each weak entity type  $W$  in the ER schema with owner entity type  $E$ , create a relation  $R$  and include all simple attributes (or simple components of composite attributes) of  $W$  as attributes of  $R$ . In addition, include as foreign key attributes of  $R$ , the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of  $W$ . The primary key of  $R$  is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type  $W$ , if any.

If there is a weak entity type  $E_2$  whose owner is also a weak entity type  $E_1$ , then  $E_1$  should be mapped before  $E_2$  to determine its primary key first.

In our example, we create the relation DEPENDENT in this step to correspond to the weak entity type DEPENDENT (see Figure 9.3(b)). We include the primary key Ssn of the EMPLOYEE relation—which corresponds to the owner entity type—as a foreign key attribute of DEPENDENT; we rename it Essn, although this is not necessary.

**Figure 9.3**

Illustration of some mapping steps.

- (a) *Entity* relations after step 1.
- (b) Additional *weak entity* relation after step 2.
- (c) *Relationship* relation after step 5.
- (d) Relation representing multivalued attribute after step 6.

**(a) EMPLOYEE**

Fname	Minit	Lname	<u>Ssn</u>	Bdate	Address	Sex	Salary
-------	-------	-------	------------	-------	---------	-----	--------

**DEPARTMENT**

Dname	<u>Dnumber</u>
-------	----------------

**PROJECT**

Pname	<u>Pnumber</u>	Plocation
-------	----------------	-----------

**(b) DEPENDENT**

<u>Essn</u>	<u>Dependent_name</u>	Sex	Bdate	Relationship
-------------	-----------------------	-----	-------	--------------

**(c) WORKS\_ON**

<u>Essn</u>	<u>Pno</u>	Hours
-------------	------------	-------

**(d) DEPT\_LOCATIONS**

<u>Dnumber</u>	<u>Dlocation</u>
----------------	------------------

The primary key of the **DEPENDENT** relation is the combination {Essn, Dependent\_name}, because Dependent\_name (also renamed from Name in Figure 9.1) is the partial key of **DEPENDENT**.

It is common to choose the propagate (**CASCADE**) option for the referential triggered action (see Section 4.2) on the foreign key in the relation corresponding to the weak entity type, since a weak entity has an existence dependency on its owner entity. This can be used for both **ON UPDATE** and **ON DELETE**.

**Step 3: Mapping of Binary 1:1 Relationship Types.** For each binary 1:1 relationship type *R* in the ER schema, identify the relations *S* and *T* that correspond to the entity types participating in *R*. There are three possible approaches: (1) the foreign key approach, (2) the merged relationship approach, and (3) the cross-reference or relationship relation approach. The first approach is the most useful and should be followed unless special conditions exist, as we discuss below.

1. **Foreign key approach:** Choose one of the relations—*S*, say—and include as a foreign key in *S* the primary key of *T*. It is better to choose an entity type with *total participation* in *R* in the role of *S*. Include all the simple attributes (or simple components of composite attributes) of the 1:1 relationship type *R* as attributes of *S*.

In our example, we map the 1:1 relationship type **MANAGES** from Figure 9.1 by choosing the participating entity type **DEPARTMENT** to serve in the role of *S* because its participation in the **MANAGES** relationship type is total (every department has a manager). We include the primary key of the **EMPLOYEE** relation as foreign key in the **DEPARTMENT** relation and rename it **Mgr\_ssn**. We also include the simple attribute **Start\_date** of the **MANAGES** relationship type in the **DEPARTMENT** relation and rename it **Mgr\_start\_date** (see Figure 9.2).

Note that it is possible to include the primary key of *S* as a foreign key in *T* instead. In our example, this amounts to having a foreign key attribute, say **Department\_managed** in the **EMPLOYEE** relation, but it will have a **NULL** value for employee tuples who do not manage a department. If only 2 percent of employees manage a department, then 98 percent of the foreign keys would be **NULL** in this case. Another possibility is to have foreign keys in both relations *S* and *T* redundantly, but this creates redundancy and incurs a penalty for consistency maintenance.

2. **Merged relation approach:** An alternative mapping of a 1:1 relationship type is to merge the two entity types and the relationship into a single relation. This is possible when *both participations are total*, as this would indicate that the two tables will have the exact same number of tuples at all times.
3. **Cross-reference or relationship relation approach:** The third option is to set up a third relation *R* for the purpose of cross-referencing the primary keys of the two relations *S* and *T* representing the entity types. As we will see, this approach is required for binary *M:N* relationships. The relation *R* is called a **relationship relation** (or sometimes a **lookup table**), because each

tuple in  $R$  represents a relationship instance that relates one tuple from  $S$  with one tuple from  $T$ . The relation  $R$  will include the primary key attributes of  $S$  and  $T$  as foreign keys to  $S$  and  $T$ . The primary key of  $R$  will be one of the two foreign keys, and the other foreign key will be a unique key of  $R$ . The drawback is having an extra relation, and requiring an extra join operation when combining related tuples from the tables.

**Step 4: Mapping of Binary 1:N Relationship Types.** For each regular binary 1:N relationship type  $R$ , identify the relation  $S$  that represents the participating entity type at the  $N$ -side of the relationship type. Include as foreign key in  $S$  the primary key of the relation  $T$  that represents the other entity type participating in  $R$ ; we do this because each entity instance on the  $N$ -side is related to at most one entity instance on the 1-side of the relationship type. Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of  $S$ .

In our example, we now map the 1:N relationship types `WORKS_FOR`, `CONTROLS`, and `SUPERVISION` from Figure 9.1. For `WORKS_FOR` we include the primary key `Dnumber` of the `DEPARTMENT` relation as foreign key in the `EMPLOYEE` relation and call it `Dno`. For `SUPERVISION` we include the primary key of the `EMPLOYEE` relation as foreign key in the `EMPLOYEE` relation itself—because the relationship is recursive—and call it `Super_ssn`. The `CONTROLS` relationship is mapped to the foreign key attribute `Dnum` of `PROJECT`, which references the primary key `Dnumber` of the `DEPARTMENT` relation. These foreign keys are shown in Figure 9.2.

An alternative approach is to use the **relationship relation** (cross-reference) option as in the third option for binary 1:1 relationships. We create a separate relation  $R$  whose attributes are the primary keys of  $S$  and  $T$ , which will also be foreign keys to  $S$  and  $T$ . The primary key of  $R$  is the same as the primary key of  $S$ . This option can be used if few tuples in  $S$  participate in the relationship to avoid excessive NULL values in the foreign key.

**Step 5: Mapping of Binary M:N Relationship Types.** For each binary M:N relationship type  $R$ , create a new relation  $S$  to represent  $R$ . Include as foreign key attributes in  $S$  the primary keys of the relations that represent the participating entity types; their *combination* will form the primary key of  $S$ . Also include any simple attributes of the M:N relationship type (or simple components of composite attributes) as attributes of  $S$ . Notice that we cannot represent an M:N relationship type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate *relationship relation*  $S$ .

In our example, we map the M:N relationship type `WORKS_ON` from Figure 9.1 by creating the relation `WORKS_ON` in Figure 9.2. We include the primary keys of the `PROJECT` and `EMPLOYEE` relations as foreign keys in `WORKS_ON` and rename them `Pno` and `Essn`, respectively. We also include an attribute `Hours` in `WORKS_ON` to represent the `Hours` attribute of the relationship type. The primary key of the `WORKS_ON` relation is the combination of the foreign key attributes  $\{\text{Essn}, \text{Pno}\}$ . This **relationship relation** is shown in Figure 9.3(c).

The propagate (CASCADE) option for the referential triggered action (see Section 4.2) should be specified on the foreign keys in the relation corresponding to the relationship  $R$ , since each relationship instance has an existence dependency on each of the entities it relates. This can be used for both ON UPDATE and ON DELETE.

Notice that we can always map 1:1 or 1:N relationships in a manner similar to M:N relationships by using the cross-reference (relationship relation) approach, as we discussed earlier. This alternative is particularly useful when few relationship instances exist, in order to avoid NULL values in foreign keys. In this case, the primary key of the relationship relation will be *only one* of the foreign keys that reference the participating entity relations. For a 1:N relationship, the primary key of the relationship relation will be the foreign key that references the entity relation on the N-side. For a 1:1 relationship, either foreign key can be used as the primary key of the relationship relation.

**Step 6: Mapping of Multivalued Attributes.** For each multivalued attribute  $A$ , create a new relation  $R$ . This relation  $R$  will include an attribute corresponding to  $A$ , plus the primary key attribute  $K$ —as a foreign key in  $R$ —of the relation that represents the entity type or relationship type that has  $A$  as a multivalued attribute. The primary key of  $R$  is the combination of  $A$  and  $K$ . If the multivalued attribute is composite, we include its simple components.

In our example, we create a relation DEPT\_LOCATIONS (see Figure 9.3(d)). The attribute Dlocation represents the multivalued attribute LOCATIONS of DEPARTMENT, while Dnumber—as foreign key—represents the primary key of the DEPARTMENT relation. The primary key of DEPT\_LOCATIONS is the combination of {Dnumber, Dlocation}. A separate tuple will exist in DEPT\_LOCATIONS for each location that a department has.

The propagate (CASCADE) option for the referential triggered action (see Section 4.2) should be specified on the foreign key in the relation  $R$  corresponding to the multivalued attribute for both ON UPDATE and ON DELETE. We should also note that the key of  $R$  when mapping a composite, multivalued attribute requires some analysis of the meaning of the component attributes. In some cases, when a multivalued attribute is composite, only some of the component attributes are required to be part of the key of  $R$ ; these attributes are similar to a partial key of a weak entity type that corresponds to the multivalued attribute (see Section 7.5).

Figure 9.2 shows the COMPANY relational database schema obtained with steps 1 through 6, and Figure 3.6 shows a sample database state. Notice that we did not yet discuss the mapping of  $n$ -ary relationship types ( $n > 2$ ) because none exist in Figure 9.1; these are mapped in a similar way to M:N relationship types by including the following additional step in the mapping algorithm.

**Step 7: Mapping of  $N$ -ary Relationship Types.** For each  $n$ -ary relationship type  $R$ , where  $n > 2$ , create a new relation  $S$  to represent  $R$ . Include as foreign key attributes in  $S$  the primary keys of the relations that represent the participating entity types. Also include any simple attributes of the  $n$ -ary relationship type (or

simple components of composite attributes) as attributes of  $S$ . The primary key of  $S$  is usually a combination of all the foreign keys that reference the relations representing the participating entity types. However, if the cardinality constraints on any of the entity types  $E$  participating in  $R$  is 1, then the primary key of  $S$  should not include the foreign key attribute that references the relation  $E'$  corresponding to  $E$  (see the discussion in Section 7.9.2 concerning constraints on  $n$ -ary relationships).

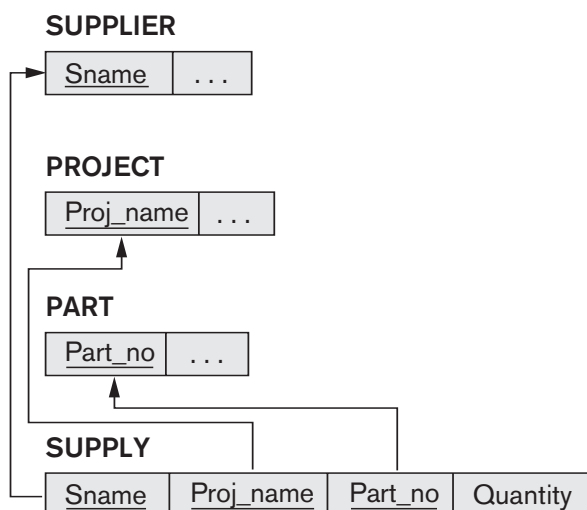
For example, consider the relationship type **SUPPLY** in Figure 7.17. This can be mapped to the relation **SUPPLY** shown in Figure 9.4, whose primary key is the combination of the three foreign keys {Sname, Proj\_name, Part\_no}.

### 9.1.2 Discussion and Summary of Mapping for ER Model Constructs

Table 9.1 summarizes the correspondences between ER and relational model constructs and constraints.

One of the main points to note in a relational schema, in contrast to an ER schema, is that relationship types are not represented explicitly; instead, they are represented by having two attributes  $A$  and  $B$ , one a primary key and the other a foreign key (over the same domain) included in two relations  $S$  and  $T$ . Two tuples in  $S$  and  $T$  are related when they have the same value for  $A$  and  $B$ . By using the **EQUIJOIN** operation (or **NATURAL JOIN** if the two join attributes have the same name) over  $S.A$  and  $T.B$ , we can combine all pairs of related tuples from  $S$  and  $T$  and materialize the relationship. When a binary 1:1 or 1:N relationship type is involved, a single join operation is usually needed. For a binary M:N relationship type, two join operations are needed, whereas for  $n$ -ary relationship types,  $n$  joins are needed to fully materialize the relationship instances.

**Figure 9.4**  
Mapping the  $n$ -ary  
relationship type  
**SUPPLY** from Figure  
7.17(a).



**Table 9.1** Correspondence between ER and Relational Models

ER MODEL	RELATIONAL MODEL
Entity type	<i>Entity</i> relation
1:1 or 1:N relationship type	Foreign key (or <i>relationship</i> relation)
M:N relationship type	<i>Relationship</i> relation and <i>two</i> foreign keys
<i>n</i> -ary relationship type	<i>Relationship</i> relation and <i>n</i> foreign keys
Simple attribute	Attribute
Composite attribute	Set of simple component attributes
Multivalued attribute	Relation and foreign key
Value set	Domain
Key attribute	Primary (or secondary) key

For example, to form a relation that includes the employee name, project name, and hours that the employee works on each project, we need to connect each EMPLOYEE tuple to the related PROJECT tuples via the WORKS\_ON relation in Figure 9.2. Hence, we must apply the EQUIJOIN operation to the EMPLOYEE and WORKS\_ON relations with the join condition  $Ssn = Essn$ , and then apply another EQUIJOIN operation to the resulting relation and the PROJECT relation with join condition  $Pno = Pnumber$ . In general, when multiple relationships need to be traversed, numerous join operations must be specified. A relational database user must always be aware of the foreign key attributes in order to use them correctly in combining related tuples from two or more relations. This is sometimes considered to be a drawback of the relational data model, because the foreign key/primary key correspondences are not always obvious upon inspection of relational schemas. If an EQUIJOIN is performed among attributes of two relations that do not represent a foreign key/primary key relationship, the result can often be meaningless and may lead to spurious data. For example, the reader can try joining the PROJECT and DEPT\_LOCATIONS relations on the condition  $Dlocation = Plocation$  and examine the result (see the discussion of spurious tuples in Section 15.1.4).

In the relational schema we create a separate relation for *each* multivalued attribute. For a particular entity with a set of values for the multivalued attribute, the key attribute value of the entity is repeated once for each value of the multivalued attribute in a separate tuple because the basic relational model does *not* allow multiple values (a list, or a set of values) for an attribute in a single tuple. For example, because department 5 has three locations, three tuples exist in the DEPT\_LOCATIONS relation in Figure 3.6; each tuple specifies one of the locations. In our example, we apply EQUIJOIN to DEPT\_LOCATIONS and DEPARTMENT on the Dnumber attribute to get the values of all locations along with other DEPARTMENT attributes. In the resulting relation, the values of the other DEPARTMENT attributes are repeated in separate tuples for every location that a department has.

The basic relational algebra does not have a NEST or COMPRESS operation that would produce a set of tuples of the form  $\{ \langle '1', 'Houston' \rangle, \langle '4', 'Stafford' \rangle, \langle '5', \{ 'Bellaire', 'Sugarland', 'Houston' \} \rangle \}$  from the DEPT\_LOCATIONS relation in Figure 3.6. This is a serious drawback of the basic normalized or *flat* version of the relational model. The object data model and object-relational systems (see Chapter 11) do allow multivalued attributes.

## 9.2 Mapping EER Model Constructs to Relations

Next, we discuss the mapping of EER model constructs to relations by extending the ER-to-relational mapping algorithm that was presented in Section 9.1.1.

### 9.2.1 Mapping of Specialization or Generalization

There are several options for mapping a number of subclasses that together form a specialization (or alternatively, that are generalized into a superclass), such as the {SECRETARY, TECHNICIAN, ENGINEER} subclasses of EMPLOYEE in Figure 8.4. We can add a further step to our ER-to-relational mapping algorithm from Section 9.1.1, which has seven steps, to handle the mapping of specialization. Step 8, which follows, gives the most common options; other mappings are also possible. We discuss the conditions under which each option should be used. We use  $\text{Attrs}(R)$  to denote *the attributes of relation R*, and  $\text{PK}(R)$  to denote *the primary key of R*. First we describe the mapping formally, then we illustrate it with examples.

**Step 8: Options for Mapping Specialization or Generalization.** Convert each specialization with  $m$  subclasses  $\{S_1, S_2, \dots, S_m\}$  and (generalized) superclass  $C$ , where the attributes of  $C$  are  $\{k, a_1, \dots, a_n\}$  and  $k$  is the (primary) key, into relation schemas using one of the following options:

- **Option 8A: Multiple relations—superclass and subclasses.** Create a relation  $L$  for  $C$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L) = k$ . Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 \leq i \leq m$ , with the attributes  $\text{Attrs}(L_i) = \{k\} \cup \{\text{attributes of } S_i\}$  and  $\text{PK}(L_i) = k$ . This option works for any specialization (total or partial, disjoint or overlapping).
- **Option 8B: Multiple relations—subclass relations only.** Create a relation  $L_i$  for each subclass  $S_i$ ,  $1 \leq i \leq m$ , with the attributes  $\text{Attrs}(L_i) = \{\text{attributes of } S_i\} \cup \{k, a_1, \dots, a_n\}$  and  $\text{PK}(L_i) = k$ . This option only works for a specialization whose subclasses are *total* (every entity in the superclass must belong to (at least) one of the subclasses). Additionally, it is only recommended if the specialization has the *disjointness constraint* (see Section 8.3.1). If the specialization is *overlapping*, the same entity may be duplicated in several relations.
- **Option 8C: Single relation with one type attribute.** Create a single relation  $L$  with attributes  $\text{Attrs}(L) = \{k, a_1, \dots, a_n\} \cup \{\text{attributes of } S_1\} \cup \dots \cup \{\text{attributes of } S_m\} \cup \{t\}$  and  $\text{PK}(L) = k$ . The attribute  $t$  is called a **type** (or



# Βάσεις Δεδομένων I (ICE1-4001)

**Ενότητα 3:** Εισαγωγή στην υλοποίηση σχεσιακών βάσεων δεδομένων με χρήση γλώσσας SQL

**(επεκτάσεις στην παρουσίαση της ενότητας 2)**

Π. Ανδρίτσος





# Περιγραφή Ενότητας

- Σκοπός της διάλεξης αυτής και κεντρικός σκοπός του μαθήματος των Βάσεων Δεδομένων είναι η παρουσίαση των απαραίτητων εννοιών ώστε οι φοιτητές να κατανοήσουν την τεχνολογία υλοποίησης βάσεων δεδομένων.
- Γίνεται παρουσίαση της υλοποίησης-προγραμματισμού απλών βάσεων δεδομένων με γλώσσα SQL.
- Η παρουσίαση του προγραμματισμού εφαρμογών βάσεων δεδομένων γίνεται με τη βοήθεια παραδειγμάτων.

# Στόχος Ενότητας

- Κύριος στόχος του μαθήματος είναι να εφοδιάσει τους φοιτητές με εισαγωγικές γνώσεις έτσι ώστε να κατανοήσουν βασικά θέματα υλοποίησης απλών βάσεων με χρήση SQL.
- **Λέξεις κλειδιά:**  
Υλοποίηση σχεσιακής βάσης δεδομένων, Structured Query Language-SQL, MySQL



# Εισαγωγή στην υλοποίηση βάσεων δεδομένων - Γλώσσα SQL

Η γλώσσα διαχείρισης βάσεων δεδομένων **SQL** περιλαμβάνει τρεις υπογλώσσες:

- 1) Γλώσσα Ορισμού Δεδομένων - DDL (Data Definition Language): CREATE TABLE – δημιουργία πίνακα, ALTER TABLE - αλλαγή στηλών πίνακα, DROP TABLE – διαγραφή πίνακα, CREATE INDEX – δημιουργία ευρετηρίου, CREATE VIEW – δημιουργία όψης κ.λπ.
- 2) Γλώσσα Χειρισμού Δεδομένων - DML (Data Manipulation Language): INSERT – εισαγωγή γραμμών, UPDATE – τροποποίηση γραμμών, DELETE – διαγραφή γραμμών, SELECT – ανάκτηση και εμφάνιση γραμμών.
- 3) Γλώσσα Ελέγχου Δεδομένων – DCL (Data Control Language): GRANT, REVOKE



# Εισαγωγή στην υλοποίηση βάσεων δεδομένων - SELECT

Πρέπει να μάθετε να προγραμματίζετε βάσεις  
δεδομένων με τις δηλώσεις (statements) της γλώσσας SQL.

Είδατε σε κάποιο βήθος έως τώρα τις δηλώσεις

CREATE TABLE, INSERT INTO.

Θα τις χρησιμοποιήσουμε για να ορίσουμε μία βάση τεσσάρων  
(4) πινάκων: emp—πίνακας με στοιχεία υπαλλήλων, dept-  
πίνακας με στοιχεία τμημάτων, project—πίνακας με στοιχεία  
έργων, assign—πίνακας που δείχνει σε ποια έργα εργάζονται οι  
υπάλληλοι.

Μετά θα εστιάσουμε κυρίως στη δήλωση SELECT.

# Εισαγωγή στη χρήση της γλώσσας SQL και στον προγραμματισμό εφαρμογών βάσεων δεδομένων

- Θα χρησιμοποιήσουμε το Σχεσιακό Σύστημα (προϊόν) Διαχείρισης Βάσεως Δεδομένων (ΣΔΒΔ) `MySQL` (<https://www.mysql.com/>) για την υλοποίηση της σχεσιακής βάσης Διεύθυνσης Προσωπικού.

# SQL

- Η γλώσσα SQL είναι μία γλώσσα διαχείρισης βάσεων δεδομένων και περιλαμβάνει τρεις υπογλώσσες:
  - τη **Γλώσσα Ορισμού Δεδομένων** - DDL (Data Definition Language),
  - τη **Γλώσσα Χειρισμού Δεδομένων** - DML (Data Manipulation Language),
  - τη **Γλώσσα Ελέγχου Δεδομένων** - DCL (Data Control Language).
- Ακολουθεί σύνταξη δηλώσεων DDL & DML και παραδείγματα

# Σχεσιακή βάση δεδομένων προσωπικού

**assign** (πίνακας που καταχωρεί ποιοί υπάλληλοι απασχολούνται σε ποιά έργα)

Empno	Projno	ptime
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

**project** (πίνακας έργων)

PROJNO	PNAME	BUDGET
100	PAYROLL	100000
200	PERSONNEL	200000
300	SALES	150000

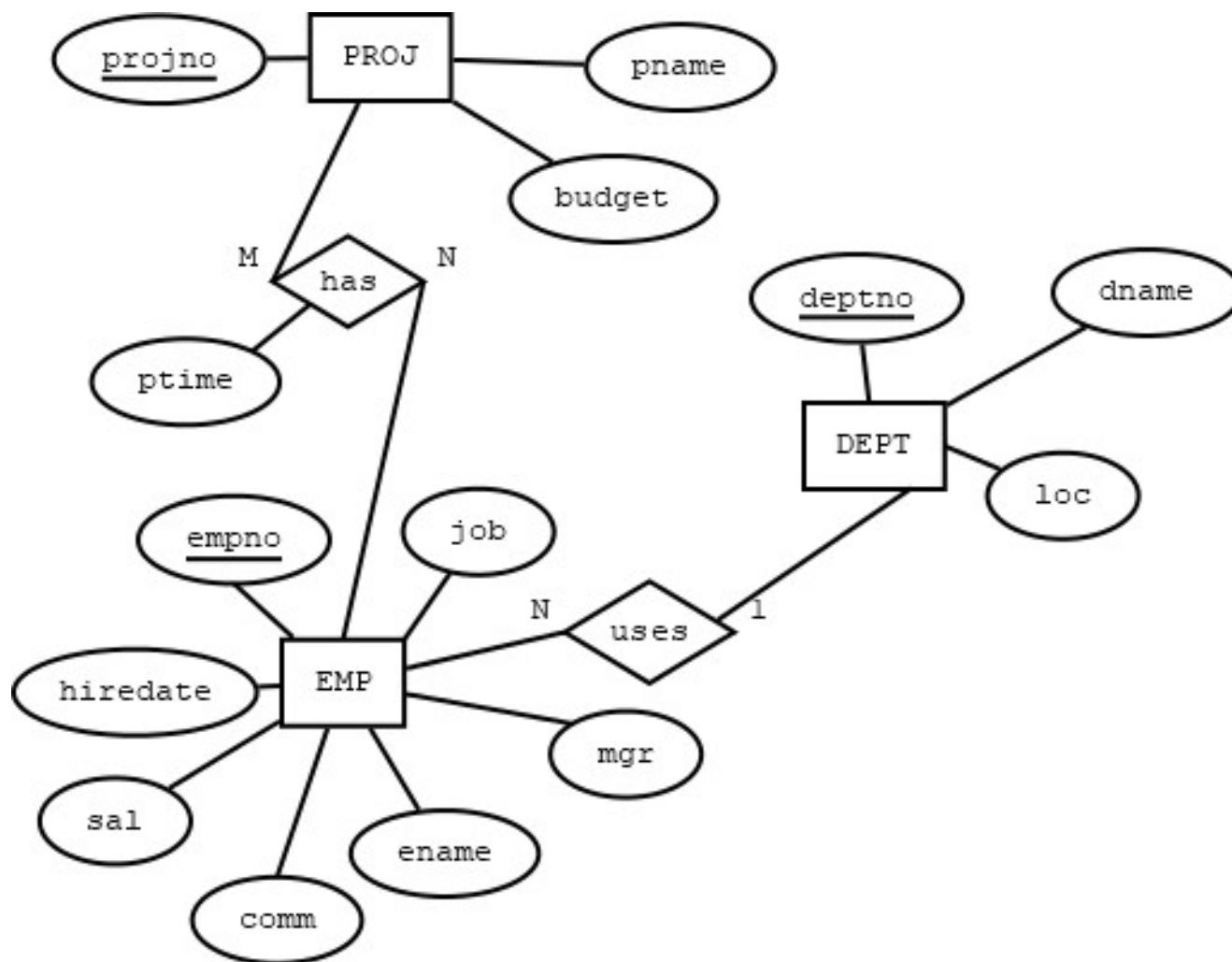
**emp** (πίνακας υπαλλήλων)

Empno	Ename	Job	Hiredate	Mgr	Sal	Comm	Deptno
10	Codd	ANALYST	1/1/89	15	3000		10
15	Elmasri	ANALYST	2/5/95	15	1200	150	10
20	Navathe	SALESMAN	7/7/77	20	2000		20
30	Date	PROGRAMMER	4/5/04	15	1800	200	10

**dept** (πίνακας τμημάτων)

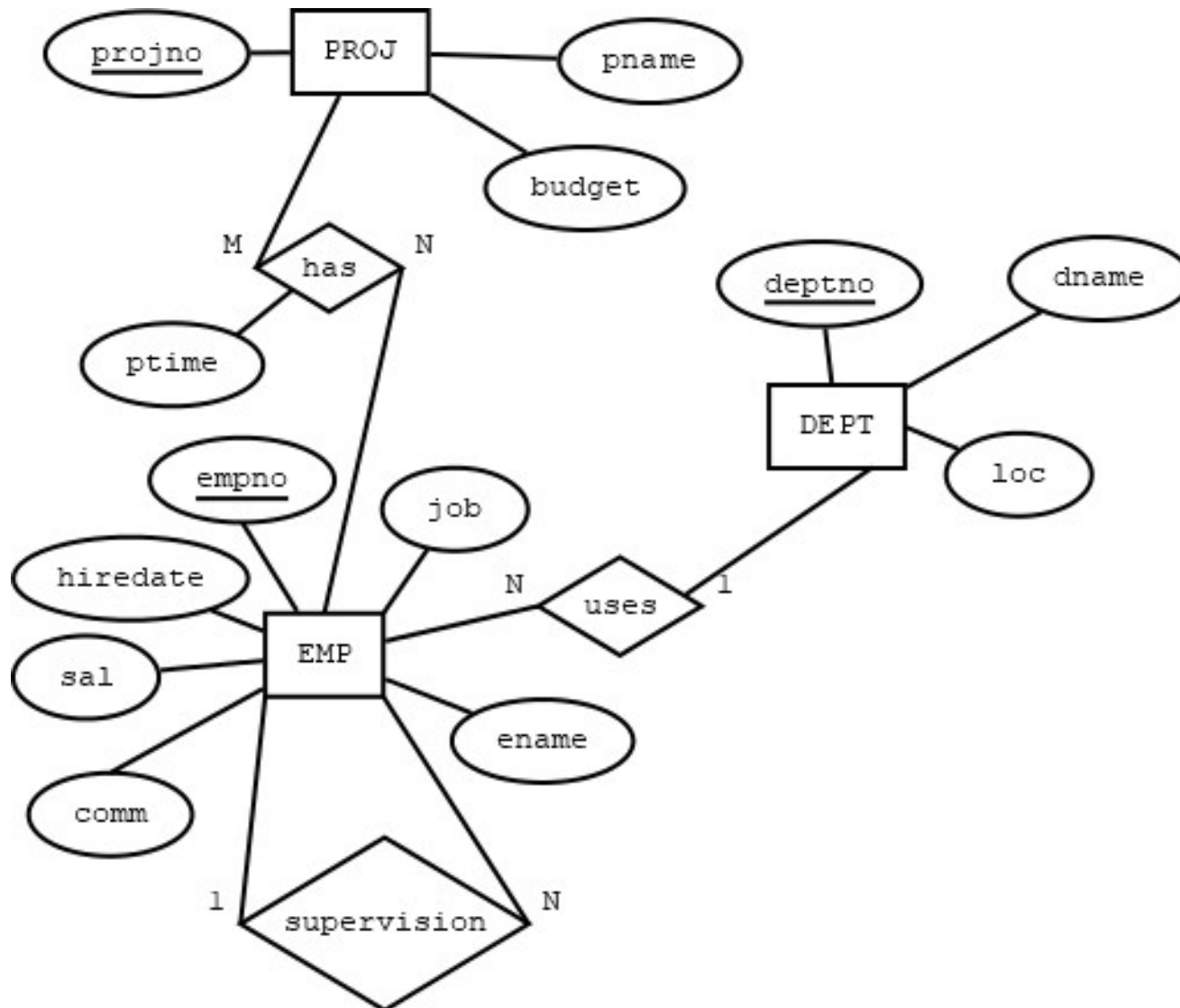
Deptno	Dname	Loc
10	ACCOUNTING	ATHENS
20	SALES	LONDON
30	RESEARCH	ATHENS
40	PAYROLL	LONDON

# Μοντέλο οντοτήτων συσχετίσεων με συμβολισμό Chen, Navathe-Elmasri κ.λπ. (συζήτηση για attribute mgr)

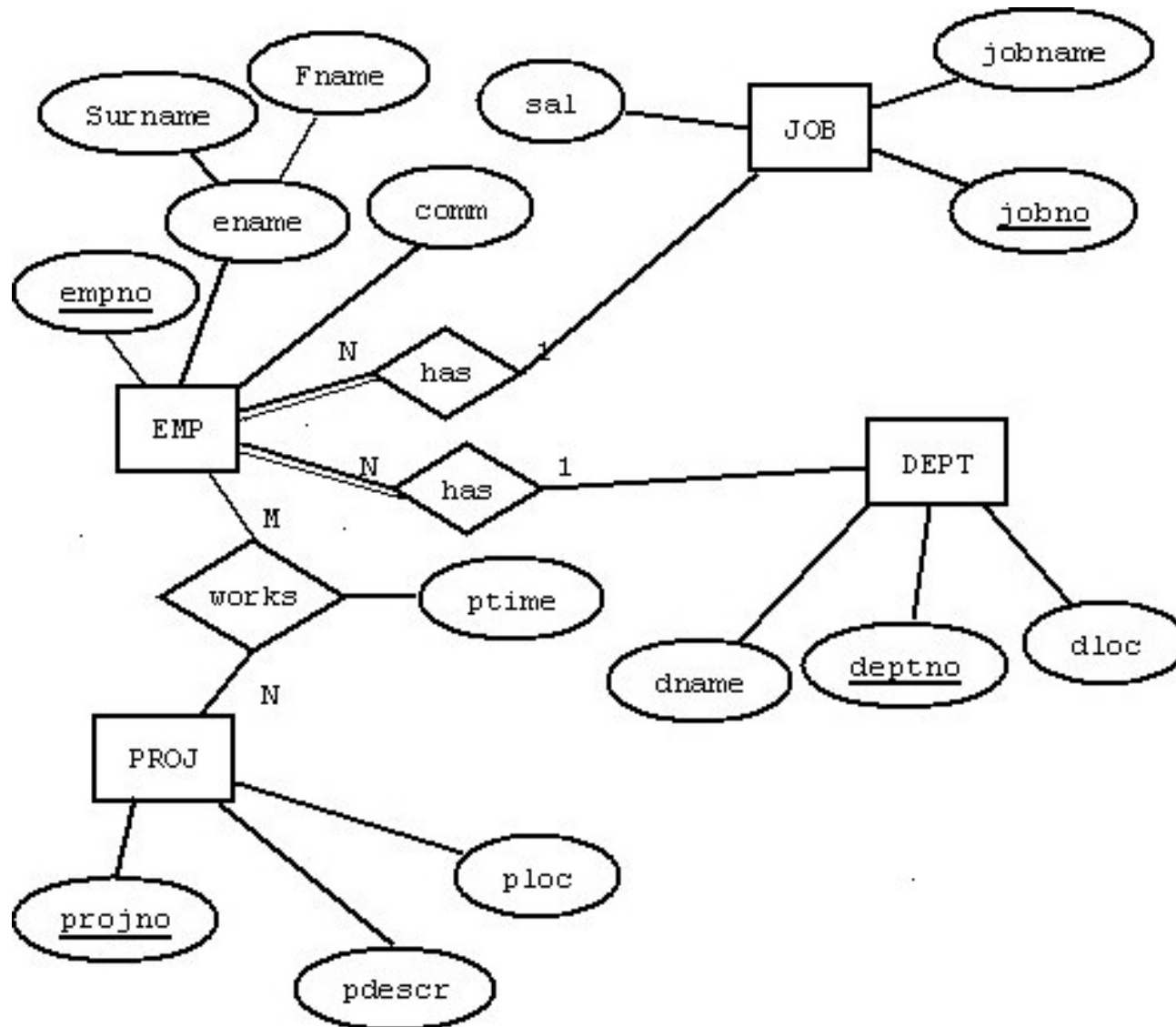





# Μοντέλο οντοτήτων συσχετίσεων με συμβολισμό Chen, Navathe-Elmasri



# Μοντέλο οντοτήτων συσχετίσεων με συμβολισμό Navathe-Elmasri (επέκταση με νέα οντότητα και άλλες μικροαλλαγές)



# Διαχείριση βάσης προσωπικού με MySQL

 MySQL 8.0 Command Line Client - Unicode

Enter password: \*\*\*\*\*

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 37

Server version: 8.0.19 MySQL Community Server - GPL

Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

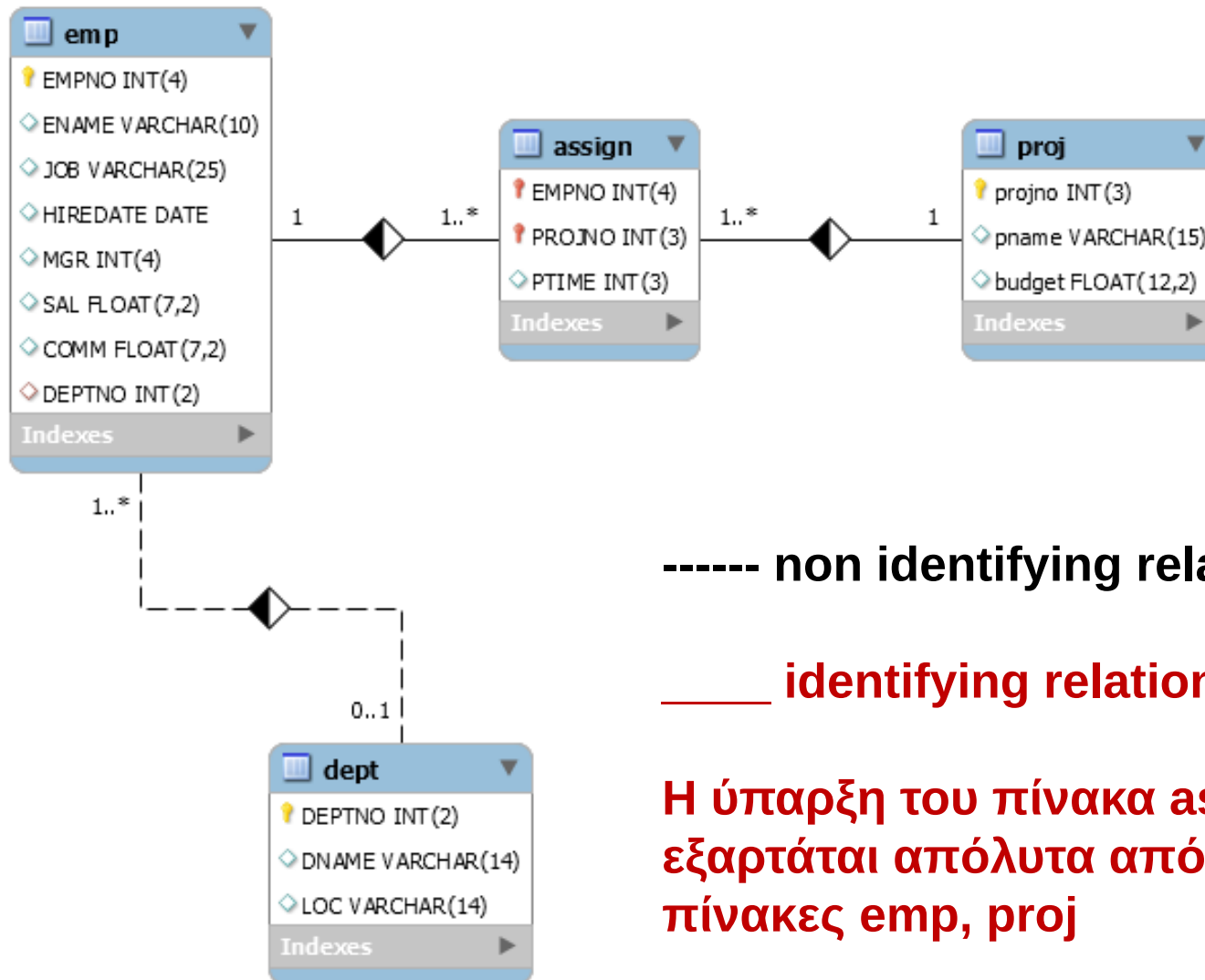
mysql>

# Δημιουργία της βάσης με **mysql**

## Οι πίνακες δημιουργούνται με κύρια και ξένα κλειδιά

```
DROP DATABASE IF EXISTS new_personnel;  
CREATE DATABASE new_personnel;  
USE new_personnel;  
  
CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,  
                    DNAME VARCHAR(14), LOC VARCHAR(14),  
                    PRIMARY KEY(DEPTNO));  
CREATE TABLE EMP(EMPNO INT(4) NOT NULL,  
                    ENAME VARCHAR(10), JOB VARCHAR(25),  
                    HIREDATE DATE, MGR INT(4), SAL FLOAT(7,2), COMM FLOAT(7,2),  
                    DEPTNO INT(2),  
                    PRIMARY KEY(EMPNO),  
                    FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));  
CREATE TABLE PROJ (projno INT(3) NOT NULL,  
                    pname VARCHAR(15),  
                    budget FLOAT(12,2),  
                    PRIMARY KEY(projno));  
CREATE TABLE ASSIGN(  
    EMPNO INT(4) NOT NULL, PROJNO INT(3) NOT NULL, PTIME INT(3),  
    PRIMARY KEY(EMPNO, PROJNO),  
    FOREIGN KEY(EMPNO) REFERENCES EMP(EMPNO),  
    FOREIGN KEY(PROJNO) REFERENCES PROJ(PROJNO));  
  
SHOW TABLES;
```

# Classic

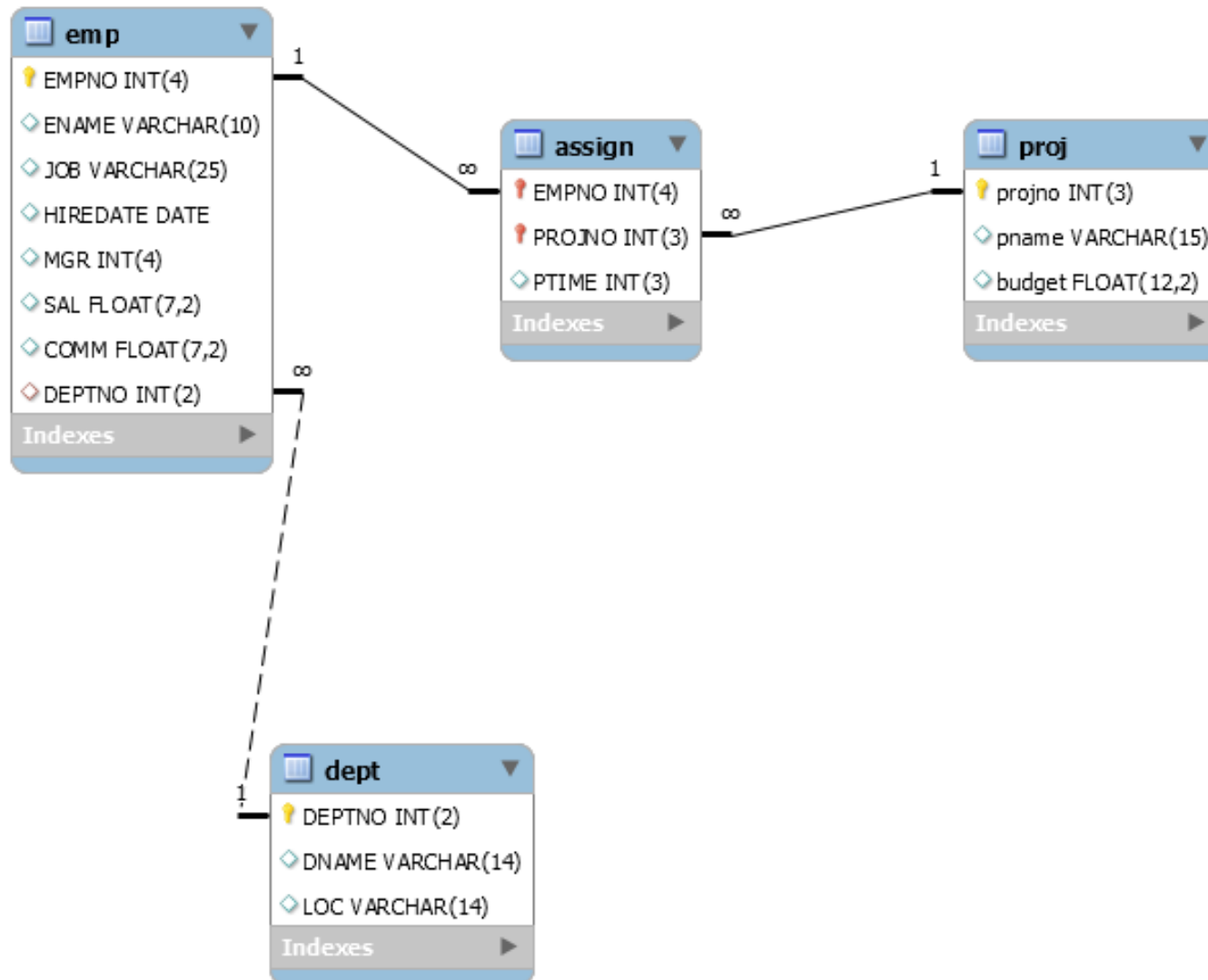


----- non identifying relationship

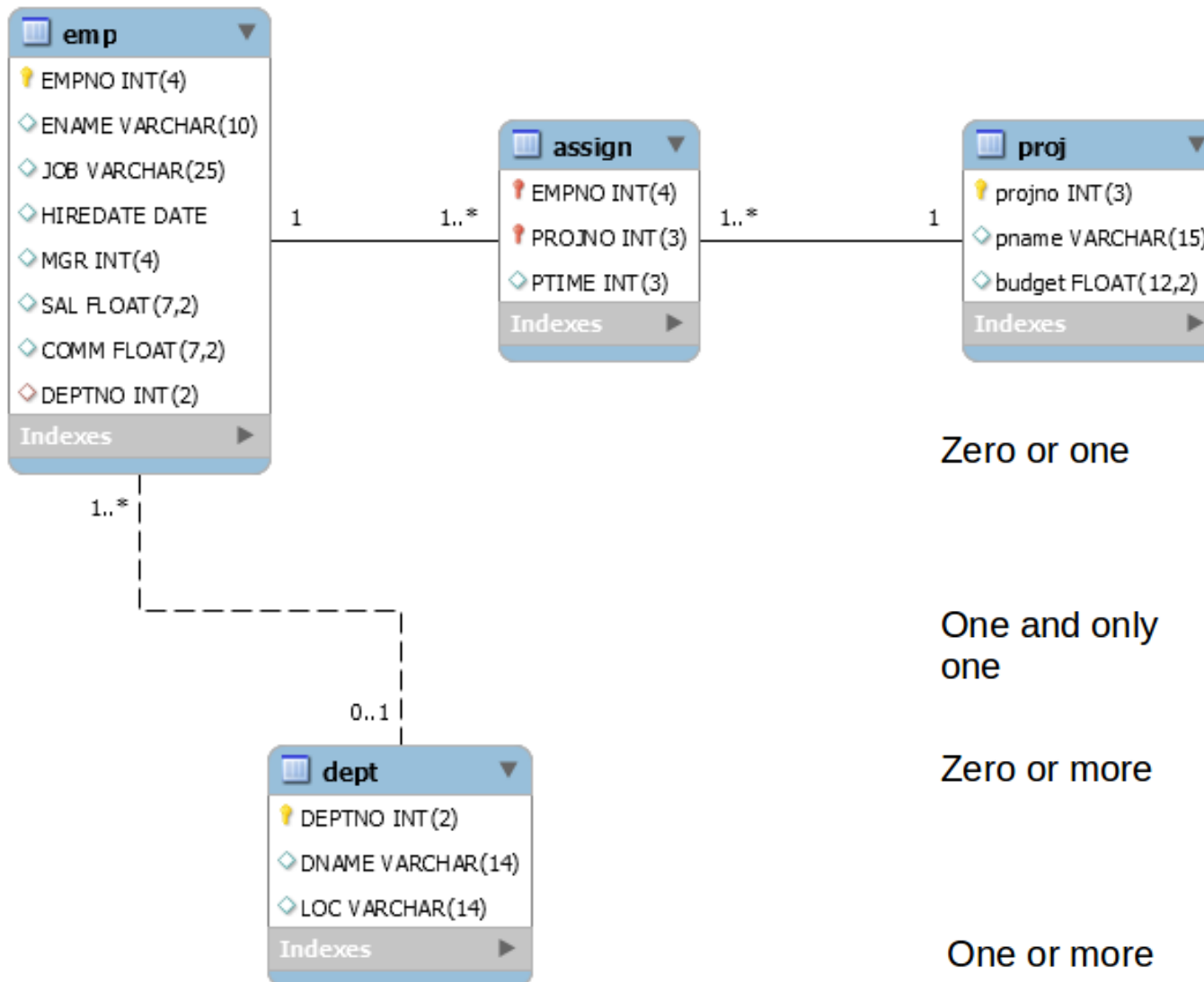
\_\_\_\_\_ identifying relationships

Η ύπαρξη του πίνακα assign εξαρτάται απόλυτα από τους πίνακες emp, proj

# Connect to columns



# UML



Zero or one

0..1

One and only one

1

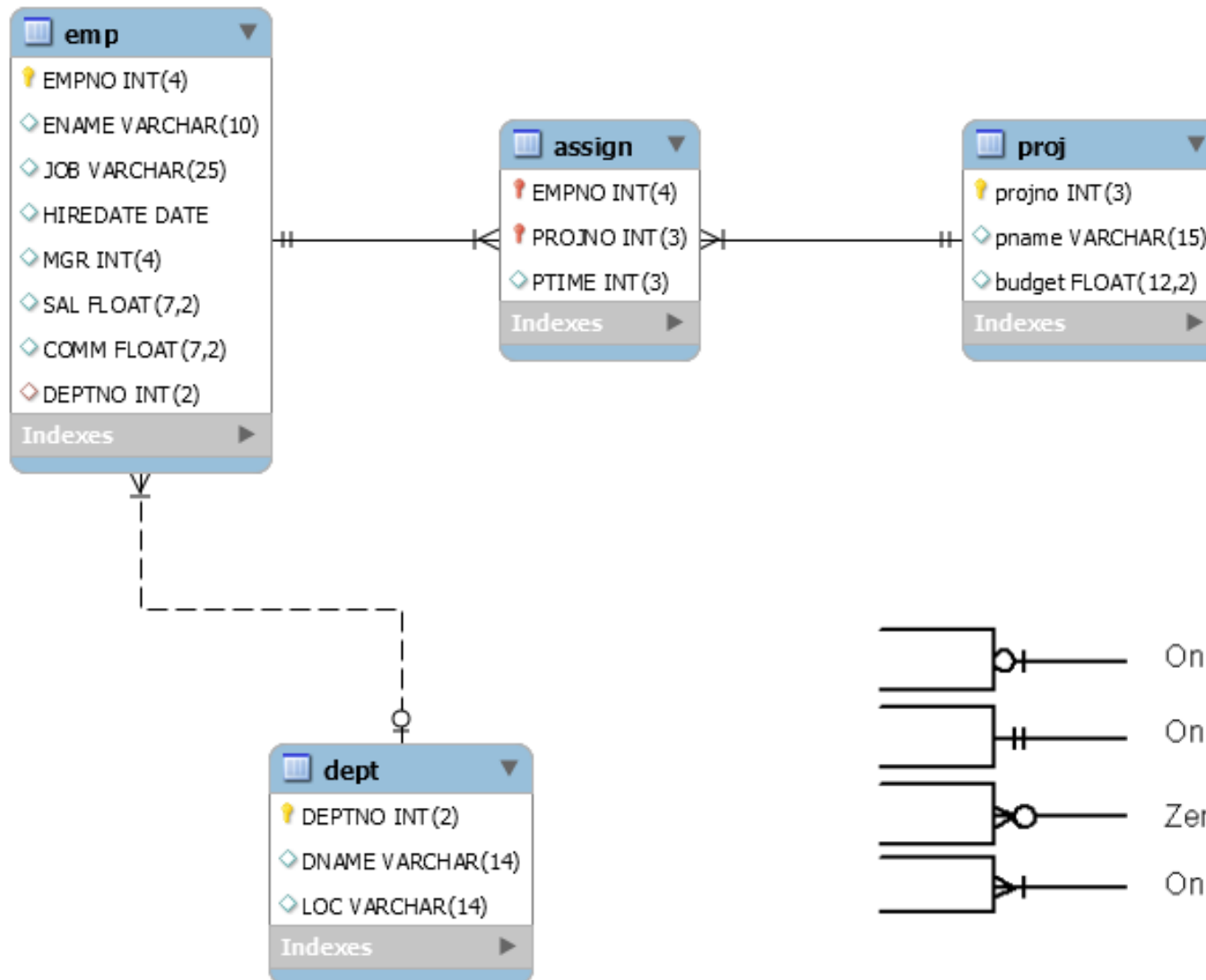
Zero or more

0..\*

One or more

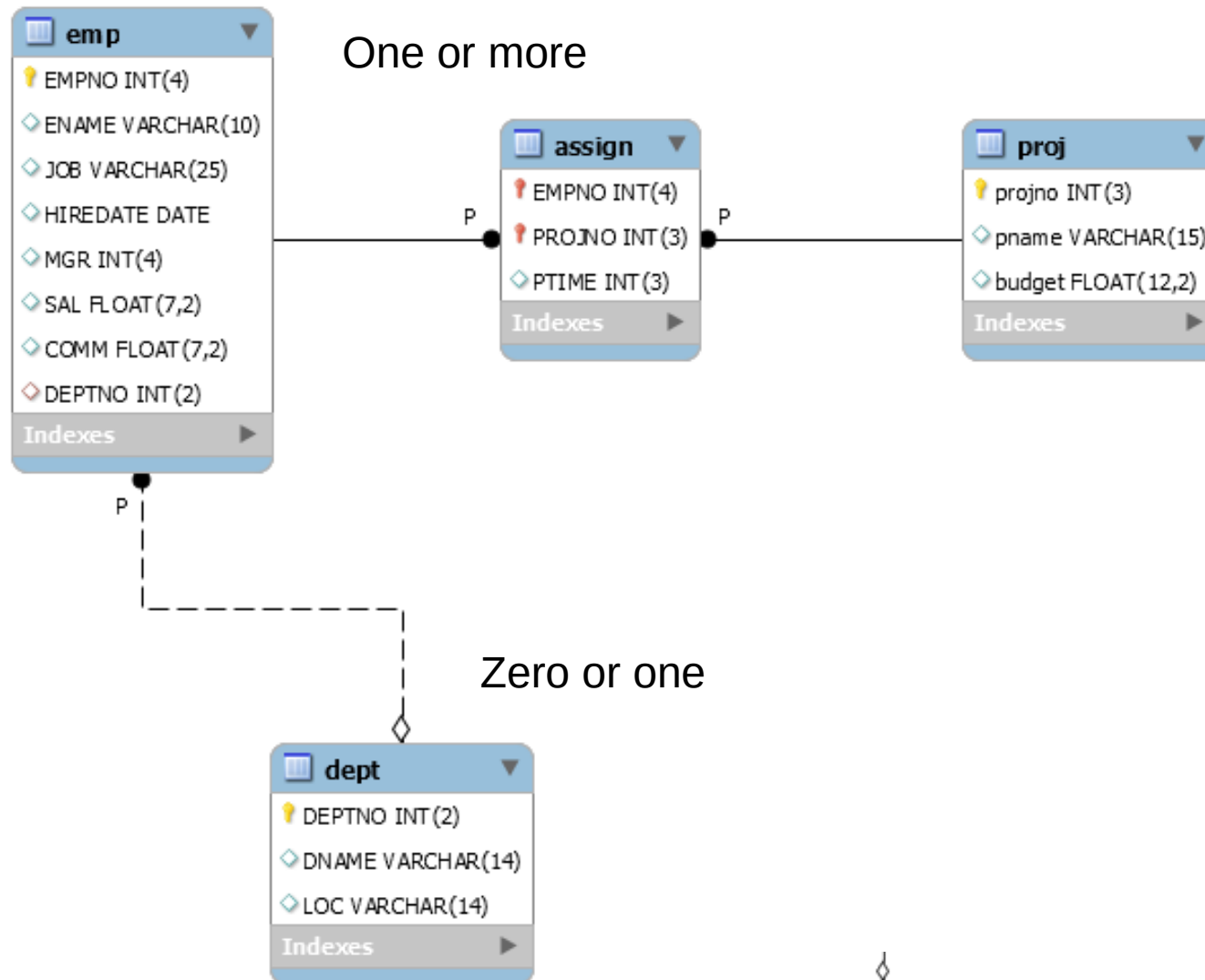
1..\*

# Crow's foot





# IDEF1X



mysql&gt; DESCRIBE DEPT;

Field	Type	Null	Key	Default	Extra
DEPTNO	int	NO	PRI	NULL	
DNAME	varchar(14)	YES		NULL	
LOC	varchar(14)	YES		NULL	

3 rows in set (0.01 sec)

mysql&gt; DESCRIBE EMP;

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
ENAME	varchar(10)	YES		NULL	
JOB	varchar(25)	YES		NULL	
HIREDATE	date	YES		NULL	
MGR	int	YES		NULL	
SAL	float(7,2)	YES		NULL	
COMM	float(7,2)	YES		NULL	
DEPTNO	int	YES	MUL	NULL	

8 rows in set (0.00 sec)

mysql&gt; DESCRIBE PROJ;

Field	Type	Null	Key	Default	Extra
projno	int	NO	PRI	NULL	
pname	varchar(15)	YES		NULL	
budget	float(12,2)	YES		NULL	

3 rows in set (0.00 sec)

mysql&gt; DESCRIBE ASSIGN;

Field	Type	Null	Key	Default	Extra
EMPNO	int	NO	PRI	NULL	
PROJNO	int	NO	PRI	NULL	
PTIME	int	YES		NULL	

3 rows in set (0.00 sec)

```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (10, 'ACCOUNTING', 'NEW YORK');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (20, 'RESEARCH', 'DALLAS');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (30, 'SALES', 'CHICAGO');
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
VALUES (40, 'OPERATIONS', 'BOSTON');
```

```
INSERT INTO EMP
VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);
INSERT INTO EMP
VALUES (15, 'ELMASRI', 'ANALYST', '1995/05/02', 15, 1200, 150, 10);
INSERT INTO EMP
VALUES (20, 'NAVATHE', 'SALESMAN', '1977/07/07', 20, 2000, NULL, 20);
INSERT INTO EMP
VALUES (30, 'DATE', 'PROGRAMMER', '2004/05/04', 15, 1800, 200, 10);
```

```
INSERT INTO proj(projno, pname, budget)
VALUES(100, 'PAYROLL', 100000);
INSERT INTO proj(projno, pname, budget)
VALUES(200, 'PERSONNEL', 200000);
INSERT INTO proj(projno, pname, budget)
VALUES(300, 'SALES', 150000);
```

```
INSERT INTO assign(empno, projno, ptime)
VALUES(10, 100, 40);
INSERT INTO assign(empno, projno, ptime)
VALUES(10, 200, 60);
INSERT INTO assign(empno, projno, ptime)
VALUES(15, 100, 100);
INSERT INTO assign(empno, projno, ptime)
VALUES(20, 200, 100);
INSERT INTO assign(empno, projno, ptime)
VALUES(30, 100, 100);
```

# Πως βλέπουμε τα δεδομένα

```
SELECT * FROM DEPT;  
SELECT * FROM EMP;  
SELECT * FROM PROJ;  
SELECT * FROM ASSIGN;
```

# Διαγραφή πινάκων

```
DROP TABLE assign;  
DROP TABLE emp;  
DROP TABLE proj;  
DROP TABLE dept;
```

# Η διαγραφή της βάσης γίνεται με τη δήλωση:

```
DROP DATABASE NEW_PERSONNEL;
```

# Βάση δεδομένων προσωπικού

SELECT \* FROM dept

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

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

SELECT \* FROM proj

PROJNO	PNAME	BUDGET
100	PAYROLL	100000
200	PERSONNEL	200000
300	SALES	150000

SELECT \* FROM assign

EMPNO	PROJNO	PTIME
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

```
mysql> SELECT * FROM DEPT;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

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

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT * FROM PROJ;
```

projno	pname	budget
100	PAYROLL	100000.00
200	PERSONNEL	200000.00
300	SALES	150000.00

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

```
mysql> SELECT * FROM ASSIGN;
```

EMPNO	PROJNO	PTIME
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

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



# Δήλωση SELECT

Συνηθισμένη σύνταξη της δήλωσης SELECT:

SELECT ονόματα στηλών ή αριθμητικές εκφράσεις κ.λπ.  
FROM όνομα πίνακα ή ονόματα πινάκων  
WHERE συνθήκη  
ORDER BY όνομα\_πίνακα.όνομα\_στήλης ASC ή DESC ή  
τίποτα, κ.λπ.

## Παράδειγμα

```
SELECT empno, ename, job, sal  
FROM emp  
WHERE job IN ('ANALYST', 'PROGRAMMER')  
AND sal >= 1000 AND sal <= 7000  
ORDER BY job, ename;
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT empno, ename, job, sal  
-> FROM emp  
-> WHERE job IN ('ANALYST', 'PROGRAMMER')  
-> AND sal >= 1000 AND sal <= 7000  
-> ORDER BY job, ename;
```

empno	ename	job	sal
10	CODD	ANALYST	3000.00
15	ELMASRI	ANALYST	1200.00
30	DATE	PROGRAMMER	1800.00

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



# Προσοχή στο null!

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT EMPNO, ENAME, JOB, SAL, COMM, SAL+COMM, SAL+IFNULL(COMM,0)  
-> FROM EMP;
```

EMPNO	ENAME	JOB	SAL	COMM	SAL+COMM	SAL+IFNULL(COMM,0)
10	CODD	ANALYST	3000.00	NULL	NULL	3000.00
15	ELMASRI	ANALYST	1200.00	150.00	1350.00	1350.00
20	NAVATHE	SALESMAN	2000.00	NULL	NULL	2000.00
30	DATE	PROGRAMMER	1800.00	200.00	2000.00	2000.00

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

# Απλά παραδείγματα στη βάση μας

```
SELECT * FROM dept
```

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

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
10	CODD	ANALYST	15	01/01/1989	3000	-	10
15	ELMASRI	ANALYST	15	02/05/1995	1200	150	10
20	NAVATHE	SALESMAN	20	07/07/1977	2000	-	20
30	DATE	PROGRAMMER	15	04/05/2004	1800	200	10

```
SELECT * FROM proj
```

PROJNO	PNAME	BUDGET
100	PAYROLL	100000
200	PERSONNEL	200000
300	SALES	150000

```
SELECT * FROM assign
```

EMPNO	PROJNO	PTIME
10	100	40
10	200	60
15	100	100
20	200	100
30	100	100

```
mysql>
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT empno, ename, job, sal
-> FROM emp;
```

empno	ename	job	sal
10	CODD	ANALYST	3000.00
15	ELMASRI	ANALYST	1200.00
20	NAVATHE	SALESMAN	2000.00
30	DATE	PROGRAMMER	1800.00

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

```
mysql> SELECT ename, job, sal, empno
-> FROM emp;
```

ename	job	sal	empno
CODD	ANALYST	3000.00	10
ELMASRI	ANALYST	1200.00	15
NAVATHE	SALESMAN	2000.00	20
DATE	PROGRAMMER	1800.00	30

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

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT empno, ename, hiredate, job, sal  
-> FROM emp  
-> ORDER BY ename;
```

empno	ename	hiredate	job	sal
10	CODD	1989-01-01	ANALYST	3000.00
30	DATE	2004-05-04	PROGRAMMER	1800.00
15	ELMASRI	1995-05-02	ANALYST	1200.00
20	NAVATHE	1977-07-07	SALESMAN	2000.00

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

```
mysql> SELECT empno, ename, hiredate, job, sal  
-> FROM emp  
-> ORDER BY job;
```

empno	ename	hiredate	job	sal
10	CODD	1989-01-01	ANALYST	3000.00
15	ELMASRI	1995-05-02	ANALYST	1200.00
30	DATE	2004-05-04	PROGRAMMER	1800.00
20	NAVATHE	1977-07-07	SALESMAN	2000.00

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

```
mysql> SELECT empno, ename, hiredate, job, sal
-> FROM emp
-> ORDER BY job;
```

empno	ename	hiredate	job	sal
10	CODD	1989-01-01	ANALYST	3000.00
15	ELMASRI	1995-05-02	ANALYST	1200.00
30	DATE	2004-05-04	PROGRAMMER	1800.00
20	NAVATHE	1977-07-07	SALESMAN	2000.00

4 rows in set (0.00 sec)

```
mysql> SELECT empno, ename, hiredate, job, sal
-> FROM emp
-> ORDER BY sal;
```

empno	ename	hiredate	job	sal
15	ELMASRI	1995-05-02	ANALYST	1200.00
30	DATE	2004-05-04	PROGRAMMER	1800.00
20	NAVATHE	1977-07-07	SALESMAN	2000.00
10	CODD	1989-01-01	ANALYST	3000.00

4 rows in set (0.00 sec)

```
mysql> SELECT empno, ename, hiredate, job, sal
-> FROM emp
-> ORDER BY hiredate;
```

empno	ename	hiredate	job	sal
20	NAVATHE	1977-07-07	SALESMAN	2000.00
10	CODD	1989-01-01	ANALYST	3000.00
15	ELMASRI	1995-05-02	ANALYST	1200.00
30	DATE	2004-05-04	PROGRAMMER	1800.00

4 rows in set (0.00 sec)

```
mysql>
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

4 rows in set (0.00 sec)

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE job='ANALYST';
```

empno	ename	job	sal
10	CODD	ANALYST	3000.00
15	ELMASRI	ANALYST	1200.00

2 rows in set (0.00 sec)

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE job='ANALYST' OR job='SALESMAN';
```

empno	ename	job	sal
10	CODD	ANALYST	3000.00
15	ELMASRI	ANALYST	1200.00
20	NAVATHE	SALESMAN	2000.00

3 rows in set (0.00 sec)

```
mysql> SELECT empno, ename, job, sal
-> FROM emp
-> WHERE job='ANALYST'
-> AND sal >= 1000;
```

empno	ename	job	sal
10	CODD	ANALYST	3000.00
15	ELMASRI	ANALYST	1200.00

2 rows in set (0.00 sec)

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT MAX(SAL), MIN(SAL), AVG(SAL), COUNT(SAL)
-> FROM EMP;
```

MAX(SAL)	MIN(SAL)	AVG(SAL)	COUNT(SAL)
3000.00	1200.00	2000.000000	4

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

```
mysql> _
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT EMPNO, LOWER(ENAME), SUBSTR(ENAME, 1,3)  
-> FROM EMP;
```

EMPNO	LOWER(ENAME)	SUBSTR(ENAME, 1,3)
10	codd	COD
15	elmasri	ELM
20	navathe	NAV
30	date	DAT

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

```
mysql>
```



```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT * FROM EMP
```

```
-> WHERE SAL>1800;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20

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

```
mysql> SELECT * FROM EMP
```

```
-> WHERE SAL>=1800;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT * FROM EMP
```

```
-> WHERE SAL<>3000;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql>
mysql> SELECT * FROM EMP
-> WHERE SAL<>3000;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> SELECT * FROM EMP
-> WHERE SAL!=3000;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

**Προσοχή στο καρτεσιανό γινόμενο και τις συνδέσεις**

# Καρτεσιανό γινόμενο

```
mysql> SELECT *  
-> FROM dept, emp;
```

DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	ACCOUNTING	NEW YORK	10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
20	RESEARCH	DALLAS	10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
30	SALES	CHICAGO	10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
40	OPERATIONS	BOSTON	10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
10	ACCOUNTING	NEW YORK	15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	RESEARCH	DALLAS	15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
30	SALES	CHICAGO	15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
40	OPERATIONS	BOSTON	15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
10	ACCOUNTING	NEW YORK	20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
20	RESEARCH	DALLAS	20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	SALES	CHICAGO	20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
40	OPERATIONS	BOSTON	20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
10	ACCOUNTING	NEW YORK	30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
20	RESEARCH	DALLAS	30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
30	SALES	CHICAGO	30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
40	OPERATIONS	BOSTON	30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

16 rows in set (0.03 sec)

SELECT \*

FROM dept, emp;

# Καρτεσιανό γινόμενο

```
mysql> SELECT empno, ename, emp.deptno, dname  
-> FROM dept, emp;
```

empno	ename	deptno	dname
10	CODD	10	ACCOUNTING
10	CODD	10	RESEARCH
10	CODD	10	SALES
10	CODD	10	OPERATIONS
15	ELMASRI	10	ACCOUNTING
15	ELMASRI	10	RESEARCH
15	ELMASRI	10	SALES
15	ELMASRI	10	OPERATIONS
20	NAVATHE	20	ACCOUNTING
20	NAVATHE	20	RESEARCH
20	NAVATHE	20	SALES
20	NAVATHE	20	OPERATIONS
30	DATE	10	ACCOUNTING
30	DATE	10	RESEARCH
30	DATE	10	SALES
30	DATE	10	OPERATIONS

16 rows in set (0.00 sec)

# Σύνδεση - join

```
mysql> SELECT *  
-> FROM dept, emp  
-> WHERE dept.deptno=emp.deptno;
```

DEPTNO	DNAME	LOC	EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	ACCOUNTING	NEW YORK	10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
10	ACCOUNTING	NEW YORK	15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	RESEARCH	DALLAS	20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
10	ACCOUNTING	NEW YORK	30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

4 rows in set (0.02 sec)

SELECT \*

FROM dept, emp

WHERE dept.deptno=emp.deptno;

# συνδέσεις

```
mysql> SELECT empno, ename, emp.deptno, dname  
-> FROM dept, emp  
-> WHERE dept.deptno=emp.deptno;
```

empno	ename	deptno	dname
10	CODD	10	ACCOUNTING
15	ELMASRI	10	ACCOUNTING
20	NAVATHE	20	RESEARCH
30	DATE	10	ACCOUNTING

```
4 rows in set (0.01 sec)
```

```
mysql> SELECT empno, ename, emp.deptno, dname  
-> FROM dept  
-> JOIN emp ON dept.deptno=emp.deptno;
```

empno	ename	deptno	dname
10	CODD	10	ACCOUNTING
15	ELMASRI	10	ACCOUNTING
20	NAVATHE	20	RESEARCH
30	DATE	10	ACCOUNTING

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

**Transactions, INSERT, UPDATE, DELETE,  
COMMIT, ROLLBACK,  
SET AUTOCOMMIT=0;**



**SET AUTOCOMMIT=0;**

UPDATE emp

SET SAL=SAL \* 1.10;

SELECT \*

FROM emp;

**ROLLBACK;**

SELECT \*

FROM emp;

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

mysql> UPDATE emp
-> SET SAL=SAL*1.10;
Query OK, 4 rows affected (0.03 sec)
Rows matched: 4  Changed: 4  Warnings: 0
```

```
mysql> SELECT *
-> FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3300.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1320.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2200.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1980.00	200.00	10

4 rows in set (0.00 sec)

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> SELECT *
-> FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

4 rows in set (0.00 sec)

```
SET AUTOCOMMIT=0;
```

```
SELECT * FROM EMP;
```

```
INSERT INTO EMP
```

```
VALUES (40, 'ULLMAN', 'ANALYST', '2014/05/04', 20,  
1800, 200, 10);
```

```
SELECT * FROM EMP;
```

```
DELETE FROM emp WHERE empno=40;
```

```
SELECT * FROM emp;
```

```
ROLLBACK;
```

```
SELECT * FROM emp;
```

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

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> INSERT INTO EMP
-> VALUES (40, 'ULLMAN', 'ANALYST', '2014/05/04', 20, 1800, 200, 10);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
40	ULLMAN	ANALYST	2014-05-04	20	1800.00	200.00	10

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

```
mysql> DELETE FROM emp WHERE empno=40;
Query OK, 1 row affected (0.15 sec)
```

```
mysql> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
SET AUTOCOMMIT=0;
INSERT INTO EMP
  VALUES (40, 'ULLMAN', 'ANALYST',
    '2014/05/04', 20, 1800, 200, 10);
COMMIT;
SELECT * FROM EMP;
DELETE FROM emp WHERE empno=40;
SELECT * FROM emp;
ROLLBACK;
SELECT * FROM emp;
```

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

mysql> INSERT INTO EMP
-> VALUES (40, 'ULLMAN', 'ANALYST', '2014/05/04', 20, 1800, 200, 10);
Query OK, 1 row affected (0.00 sec)
```

```
mysql> COMMIT;
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> SELECT * FROM EMP;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
40	ULLMAN	ANALYST	2014-05-04	20	1800.00	200.00	10

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

```
mysql> DELETE FROM emp WHERE empno=40;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10

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

```
mysql> ROLLBACK;
Query OK, 0 rows affected (0.04 sec)
```

```
mysql> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	HIREDATE	MGR	SAL	COMM	DEPTNO
10	CODD	ANALYST	1989-01-01	15	3000.00	NULL	10
15	ELMASRI	ANALYST	1995-05-02	15	1200.00	150.00	10
20	NAVATHE	SALESMAN	1977-07-07	20	2000.00	NULL	20
30	DATE	PROGRAMMER	2004-05-04	15	1800.00	200.00	10
40	ULLMAN	ANALYST	2014-05-04	20	1800.00	200.00	10

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

# Αναζητήσεις που περιλαμβάνουν υπολογισμούς, πράξεις σε strings κλπ. (MySQL)

Για να υπολογίσεις αθροίσματα, γινόμενα, μέσους όρους κλπ. μπορείς να χρησιμοποιήσεις κατά την επιλογή στηλών:

- 1) αριθμητικούς τελεστές (+, -, \*, /),
- 2) συναρτήσεις (όπως IFNULL-Null Value, POWER-ύψωση σε δύναμη, ROUND-στρογγύλευση, TRUNCATE-αποκοπή, SUBSTR/SUBSTRING-substring, UPPER-μεταγραφή σε κεφαλαία, AVG-μέσος όρος, MAX-μέγιστη τιμή. MIN-ελάχιστη τιμή, SUM-άθροισμα τιμών, COUNT-πόσες είναι οι τιμές κλπ.)
- 3) παρενθέσεις,
- 4) τελεστή DISTINCT

Οι συναρτήσεις συχνά διαφέρουν από προϊόν σε προϊόν:

IFNULL (MySQL) vs NVL-Null VaLue (ORACLE)

# Πως σχηματίζουμε σύνθετες συνθήκες σε υποπρόταση WHERE

## Τελεστές σύγκρισης, Αριθμητικοί, Boole, LIKE, NOT LIKE, BETWEEN ...AND, NOT BETWEEN ... AND, σύνολα (IN)

Για να σχηματίσουμε συνθήκες μπορούμε να χρησιμοποιήσουμε:

- 1) τελεστές σύγκρισης (>, <, >=, <=, !=, <>), όπου οι δύο τελευταίοι (ή και άλλοι ανάλογα με το προϊόν) συμβολίζουν το διάφορο.
- 2) αριθμητικούς (+, -, /, \*),
- 3) τελεστές Boole (AND, OR, NOT),
- 4) τελεστή LIKE ή NOT LIKE (π.χ. ENAME NOT LIKE '%ΑΣ'),
- 5) τελεστές BETWEEN ...AND ή NOT BETWEEN ...AND  
π.χ., SAL BETWEEN 2500 AND 3000
- 6) σύνολα τιμών (τελεστής IN)  
π.χ., JOB IN ('ΠΩΛΗΤΗΣ', 'ΑΝΑΛΥΤΗΣ ΣΥΣΤΗΜΑΤΩΝ'),
- 7) παρενθέσεις.



# Πως σχηματίζουμε συνθήκες που περιλαμβάνουν υποαναζήτηση (SELECT)

Για το σχηματισμό συνθήκης με υποαναζήτηση μπορείς να χρησιμοποιήσεις:

- 1) τελεστές σύγκρισης (>, <, >=, <=, =, !=, <> )
- 2) τελεστές Boole (AND, OR, NOT)
- 3) παρενθέσεις
- 4) τελεστές ANY, ALL, IN, EXIST
- 7) συναρτήσεις.

# Αναζήτηση στοιχείων με διάταξη των αποτελεσμάτων - υποπρόταση ORDER

Η συνηθισμένη μορφή σύνταξης της δήλωσης SELECT που ταξινομεί τα αποτελέσματα είναι:

```
SELECT  ονόματα στηλών ή αριθμητικές εκφράσεις των στηλών ή ..  
FROM    όνομα πίνακα ή ονόματα πινάκων ή ....  
WHERE   συνθήκη  
ORDER  BY  όνομα_πίνακα.όνομα_στήλης ASC ή DESC ή τίποτα,  
         όνομα_πίνακα.όνομα_στήλης κ.λπ.;
```



# Πίνακες στους οποίους θα δοκιμάσουμε τις δηλώσεις SELECT

Στον πίνακα emp καταχωρίζουμε τα στοιχεία του υπαλλήλου. Στήλες του Πίνακα emp:

Empno-κωδικός υπαλλήλου, ename-όνομα υπαλλήλου, job-θέση υπαλλήλου, mgr-ο επικεφαλής του υπαλλήλου, hiredate-ημερομηνία πρόσληψης, sal-μισθός, comm-προμήθεια, deptno-κωδικός τμήματος στο οποίο εργάζεται ο υπάλληλος

Στον πίνακα dept καταχωρίζουμε τα στοιχεία των τμημάτων. Στήλες του Πίνακα dept:

Deptno-κωδικός τμήματος,  
dname-όνομα τμήματος,  
loc-έδρα τμήματος

```
DROP DATABASE IF EXISTS personnel;

CREATE DATABASE personnel;


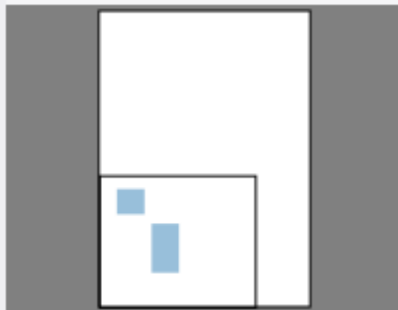

USE personnel;

CREATE TABLE DEPT(DEPTNO INT(2) NOT NULL,
                   DNAME VARCHAR(14), LOC VARCHAR(14),
                   PRIMARY KEY(DEPTNO));

CREATE TABLE EMP(EMPNO INT(4) NOT NULL,
                  ENAME VARCHAR(10), JOB VARCHAR(9),
                  MGR INT(4), HIREDATE DATE,
                  SAL FLOAT(7,2), COMM FLOAT(7,2),
                  DEPTNO INT(2),
                  PRIMARY KEY(EMPNO),
                  FOREIGN KEY(DEPTNO) REFERENCES DEPT(DEPTNO));
```



Bird's Eye

Zoom: 100%  

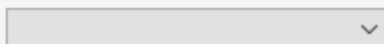
Catalog Tree

- ▼ my\_first\_db
  - Tables
  - Views
  - Routine Groups

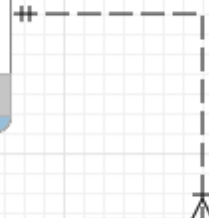
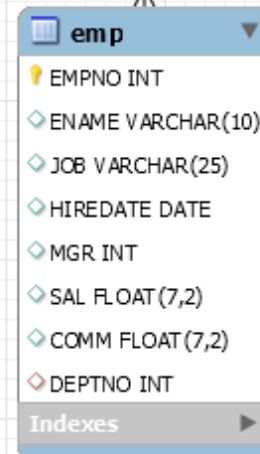
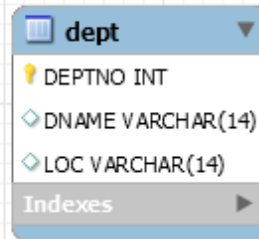


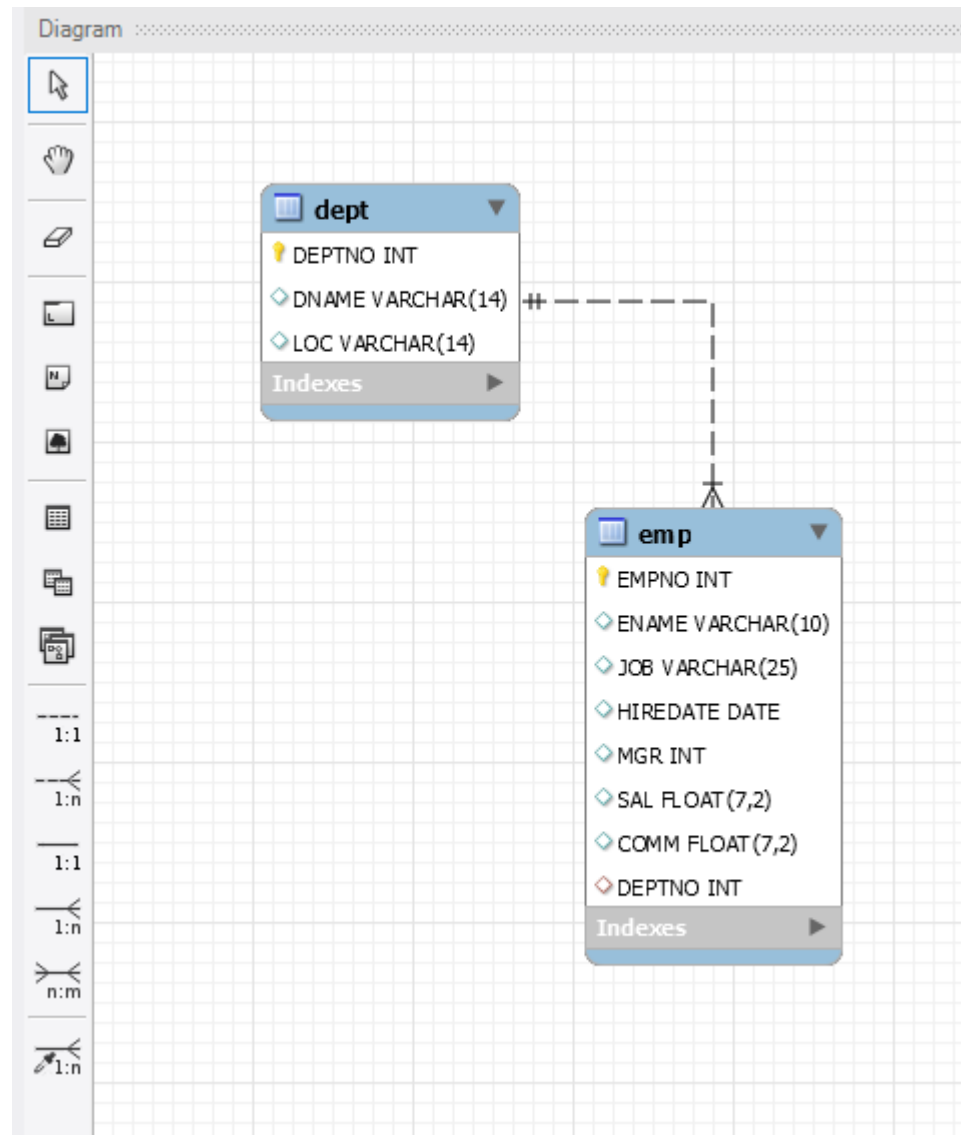
Catalog Layers User Types

Description Editor

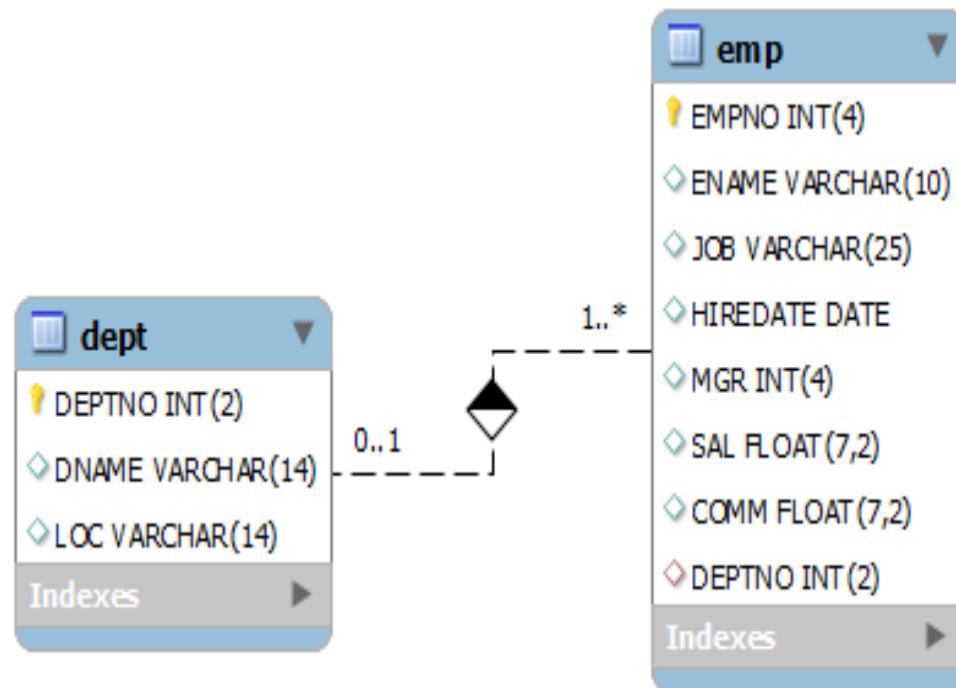


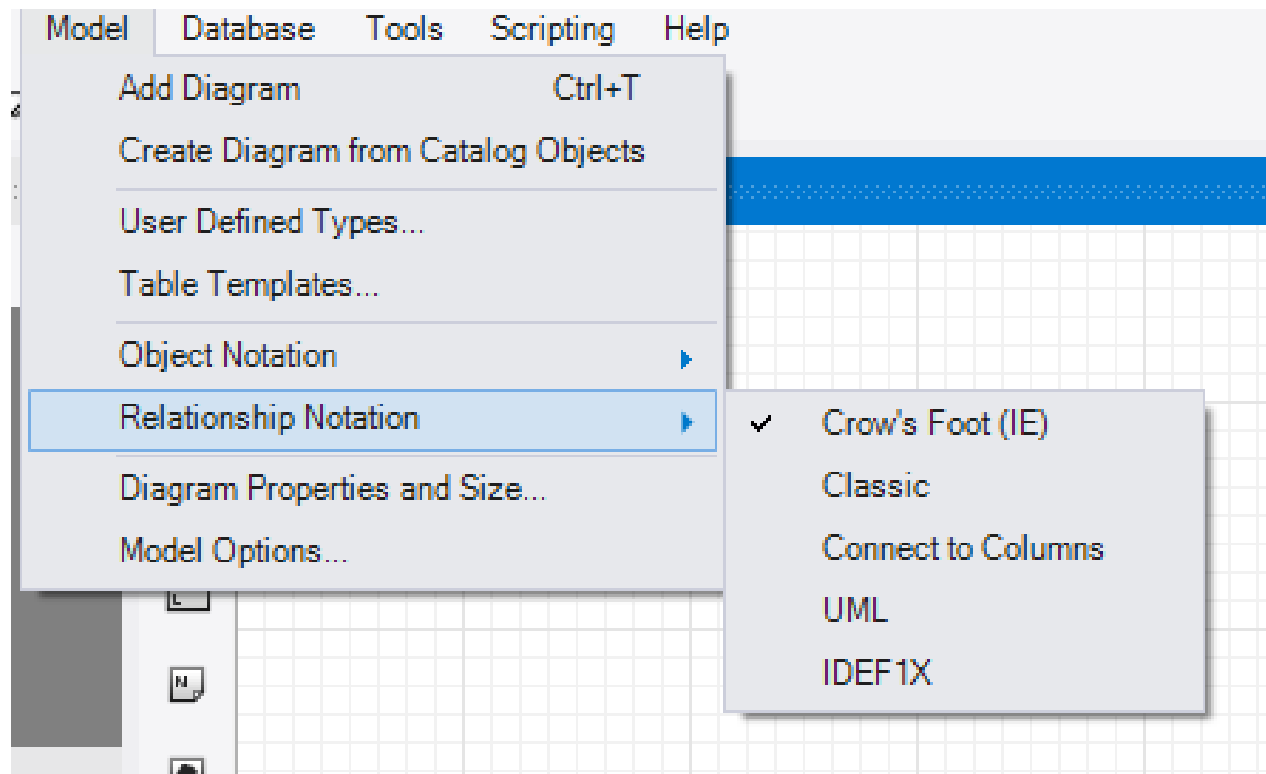
Diagram





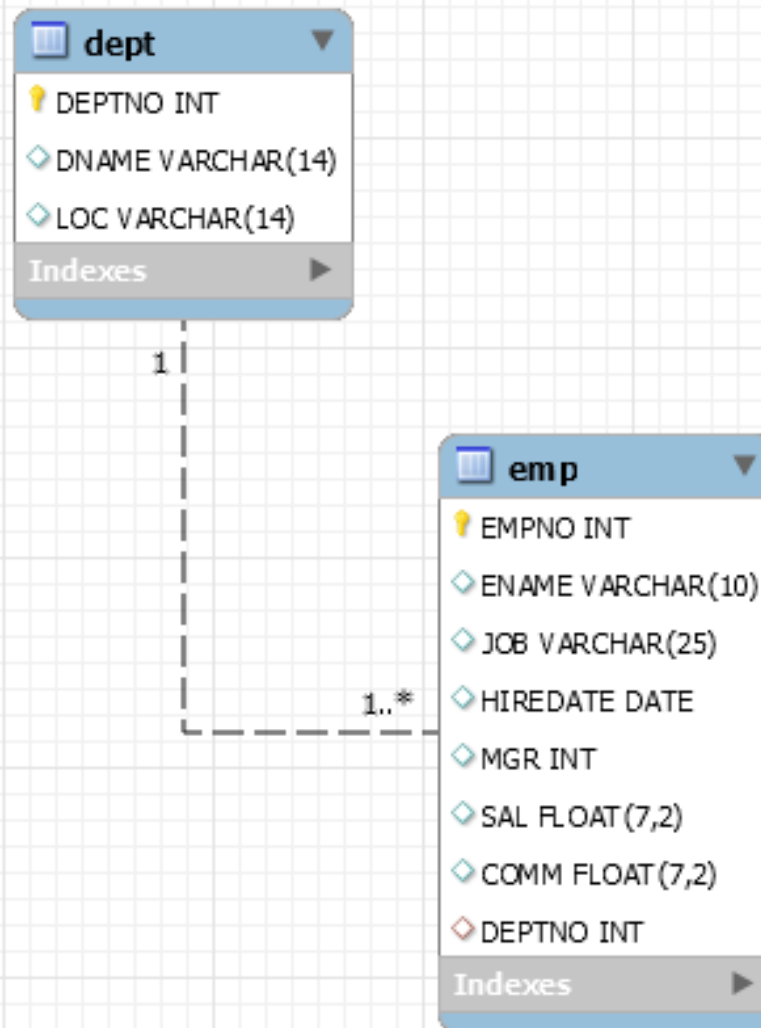
# Μοντέλο βάσης δεδομένων Client MySQL Workbench







# UML



```
INSERT INTO DEPT(DEPTNO, DNAME, LOC)
```

```
VALUES (10, 'ACCOUNTING', 'NEW YORK');
```

```
INSERT INTO DEPT VALUES (20, 'RESEARCH', 'DALLAS');
```

```
INSERT INTO DEPT(DNAME, DEPTNO, LOC)
```

```
VALUES ('SALES', 30, 'CHICAGO');
```

```
INSERT INTO DEPT(DEPTNO, LOC, DNAME)
```

```
VALUES (40, 'BOSTON', 'OPERATIONS');
```

```
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7369, 'SMITH', 'CLERK', 7902, '1980/12/17', 800, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7499, 'ALLEN', 'SALESMAN', 7698, '1981/02/20', 1600, 300, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7521, 'WARD', 'SALESMAN', 7698, '2002/02/01', 1250, 500, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7566, 'JONES', 'MANAGER', 7839, '1981/12/24', 2975, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7654, 'MARTIN', 'SALESMAN', 7698, '1981/10/28', 1250, 1400, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7698, 'BLAKE', 'MANAGER', 7839, '2001/05/02', 2850, NULL, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7782, 'CLARK', 'MANAGER', 7839, '1981/11/27', 2450, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
VALUES (7788, 'SCOTT', 'ANALYST', 7566, '1987/04/29', 3000, NULL, 20);
```

```
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7839, 'KING', 'PRESIDENT', NULL, '1987/11/12', 5000, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7844, 'TURNER', 'SALESMAN', 7698, '2007/10/19', 1500, 0, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7876, 'ADAMS', 'CLERK', 7788, '2003/05/07', 1100, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7900, 'JAMES', 'CLERK', 7698, '2003/12/12', 950, NULL, 30);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7902, 'FORD', 'ANALYST', 7566, '2003/12/19', 3000, NULL, 20);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7934, 'MILLER', 'CLERK', 7782, '2003/01/19', 1300, NULL, 10);
INSERT INTO EMP(EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO)
    VALUES (7999, 'BATES', 'ANALYST', 7566, '2004/01/04', 1300, NULL, NULL);
```

# Η βάση δεδομένων

```
mysql> SELECT * FROM DEPT;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

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

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql>
```



# Δοκιμάστε τις δηλώσεις SELECT

## Ποιό το αποτέλεσμα;

Στις επόμενες δηλώσεις SELECT δείχνουμε τους Πίνακες της βάσης στους οποίους θα δοκιμάσουμε την εντολή. Προσπαθήστε να προβλέψετε το αποτέλεσμα.

# Σύνταξη απλής δήλωσης SELECT

Να γράψετε απλές δηλώσεις - εντολές SELECT που θα βασίζονται σε ένα πίνακα.

Στις δηλώσεις αυτές θα χρησιμοποιήσετε στοιχεία που φαίνονται στην παρακάτω γενική μορφή:

**SELECT** απλές στήλες, τίτλους στα αποτελέσματα,  
πράξεις μεταξύ στηλών, συναρτήσεις, τελεστή **DISTINCT**  
**FROM** όνομα ενός πίνακα  
**WHERE** συνθήκη

Οπού η συνθήκη ανάλογα και με τη δήλωση θα περιλαμβάνει **παρενθέσεις**, τελεστές Boole (**AND, OR, NOT**), **BETWEEN ... AND, LIKE**, τελεστές σύγκρισης (**>, <, >=, <=, <>**), φωλιασμένη αναζήτηση (**SELECT ... SELECT**), **ORDER BY** κριτήρια ταξινόμησης;



# Προσπαθήστε να προβλέψετε το αποτέλεσμα

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	MANAGER	7839	01-MAY-81	2850		30
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7788	SCOTT	ANALYST	7566	19-APR-87	3000		20
7839	KING	PRESIDENT		17-NOV-81	5000		10
7844	TURNER	SALESMAN	7698	08-SEP-81	1500	0	30
7876	ADAMS	CLERK	7788	23-MAY-87	1100		20
7900	JAMES	CLERK	7698	03-DEC-81	950		30
7902	FORD	ANALYST	7566	03-DEC-81	3000		20
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7999	BATES	ANALYST	7566	23-JAN-04	1300		

```
SELECT empno, ename, job, sal  
FROM EMP  
WHERE job IN ('ANALYST', 'PROGRAMMER');
```





# Αποτελέσματα εντολής SELECT

```
SELECT * FROM emp;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7369	SMITH	CLERK	7902	17 DEC 80	800		20
7499	ALLEN	SALESMAN	7698	20 FEB 81	1600	300	30
7521	WARD	SALESMAN	7698	22 FEB 81	1250	500	30
7566	JONES	MANAGER	7839	02 APR 81	2975		20
7654	MARTIN	SALESMAN	7698	28 SEP 81	1250	1400	30
7698	BLAKE	MANAGER	7839	01 MAY 81	2850		30
7782	CLARK	MANAGER	7839	09 JUN 81	2450		10
7788	SCOTT	ANALYST	7566	19 APR 87	3000		20
7839	KING	PRESIDENT		17 NOV 81	5000		10
7844	TURNER	SALESMAN	7698	08 SEP 81	1500	0	30
7876	ADAMS	CLERK	7788	23 MAY 87	1100		20
7900	JAMES	CLERK	7698	03 DEC 81	950		30
7902	FORD	ANALYST	7566	03 DEC 81	3000		20
7934	MILLER	CLERK	7782	23 JAN 82	1300		10
7999	BATES	ANALYST	7566	23 JAN 04	1300		

```
SELECT empno, ename, job, sal  
FROM EMP  
WHERE job IN ('ANALYST', 'PROGRAMMER');
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT empno, ename, job, sal  
-> FROM EMP  
-> WHERE job IN ('ANALYST', 'PROGRAMMER');
```

empno	ename	job	sal
7788	SCOTT	ANALYST	3000.00
7902	FORD	ANALYST	3000.00
7999	BATES	ANALYST	1300.00

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



Προσπαθήστε να προβλέψετε το αποτέλεσμα στις επόμενες δηλώσεις **SELECT**

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

15 rows in set (0.00 sec)

```
SELECT empno "κωδικός",ename "όνομα",job "θέση", sal "αμοιβή",  
       sal+IFNULL(comm, 0) "σύνολο"
```

```
FROM EMP
```

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

```
ORDER BY job, ename;
```

```
SELECT empno "κωδικός",ename "όνομα",job "θέση", sal "αμοιβή",  
       sal+IFNULL(comm, 0) "σύνολο"
```

```
FROM EMP
```

```
WHERE job="ANALYST" OR JOB= "PROGRAMMER"
```

```
ORDER BY job, ename;
```

```
mysql> SELECT empno "κωδικός",ename "όνομα",job "θέση", sal "αμοιβή",
->          sal+IFNULL(comm, 0) "σύνολο"
-> FROM EMP
-> WHERE job IN ('ANALYST', 'PROGRAMMER')
-> ORDER BY job, ename;
```

κωδικός	όνομα	θέση	αμοιβή	σύνολο
7999	BATES	ANALYST	1300.00	1300.00
7902	FORD	ANALYST	3000.00	3000.00
7788	SCOTT	ANALYST	3000.00	3000.00

3 rows in set (0.00 sec)

```
mysql> SELECT empno "κωδικός",ename "όνομα",job "θέση", sal "αμοιβή",
->          sal+IFNULL(comm, 0) "σύνολο"
-> FROM EMP
-> WHERE job="ANALYST" OR JOB="PROGRAMMER"
-> ORDER BY job, ename;
```

κωδικός	όνομα	θέση	αμοιβή	σύνολο
7999	BATES	ANALYST	1300.00	1300.00
7902	FORD	ANALYST	3000.00	3000.00
7788	SCOTT	ANALYST	3000.00	3000.00

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal),  
SUM((sal+IFNULL(comm,0))), COUNT(sal), COUNT(comm), COUNT(*)  
FROM EMP;
```

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal), SUM((sal+IFNULL(comm,0))), COUNT(sal), COUNT(comm), COUNT(*)  
-> FROM EMP;
```

AVG(sal)	MIN(sal)	MAX(sal)	SUM(sal)	COUNT(sal)	SUM((sal+IFNULL(comm,0)))	COUNT(sal)	COUNT(comm)	COUNT(*)
2021.666667	800.00	5000.00	30325.00	15	32525.00	15	4	15

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

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

15 rows in set (0.00 sec)

```
SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal), SUM(sal),  
        COUNT(sal), COUNT(*)  
FROM EMP  
WHERE job="ANALYST";
```



```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal), SUM(sal),  
-> COUNT(sal), COUNT(*)  
-> FROM EMP  
-> WHERE job="ANALYST";
```

AVG(sal)	MIN(sal)	MAX(sal)	SUM(sal)	COUNT(sal)	SUM(sal)	COUNT(sal)	COUNT(*)
2433.333333	1300.00	3000.00	7300.00	3	7300.00	3	3

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

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal), SUM(sal),  
COUNT(sal), COUNT(comm), COUNT(*)  
FROM EMP  
WHERE job= "ANALYST" OR job="SALESMAN";
```

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT AVG(sal), MIN(sal), MAX(sal), SUM(sal), COUNT(sal), SUM(comm), COUNT(comm), COUNT(*)  
-> FROM EMP  
-> WHERE job="ANALYST" OR job="SALESMAN";
```

AVG(sal)	MIN(sal)	MAX(sal)	SUM(sal)	COUNT(sal)	SUM(comm)	COUNT(comm)	COUNT(*)
1842.857143	1250.00	3000.00	12900.00	7	12900.00	7	7

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

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
SELECT DISTINCT job  
FROM EMP  
ORDER BY job;
```

```
SELECT DISTINCT deptno, job  
FROM EMP  
ORDER BY deptno, job;
```

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT DISTINCT job  
-> FROM EMP  
-> ORDER BY job;
```

job
ANALYST
CLERK
MANAGER
PRESIDENT
SALESMAN

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

```
mysql>
```

```
mysql>
```

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT DISTINCT deptno, job
```

```
-> FROM EMP
```

```
-> ORDER BY deptno, job;
```

deptno	job
NULL	ANALYST
10	CLERK
10	MANAGER
10	PRESIDENT
20	ANALYST
20	CLERK
20	MANAGER
30	CLERK
30	MANAGER
30	SALESMAN

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

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

15 rows in set (0.00 sec)

```
SELECT empno, ename, job, sal  
FROM EMP  
WHERE job IN ("ANALYST", "PROGRAMMER")  
AND sal >= 1000 AND sal <= 7000;
```

```
SELECT empno, ename, job, sal  
FROM EMP  
WHERE (job IN ("ANALYST", "PROGRAMMER"))  
AND (sal >= 1300 OR sal + IFNULL(comm, 0) >= 1800)  
ORDER BY job, ename, sal;
```



```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
mysql> SELECT empno, ename, job, sal  
-> FROM EMP  
-> WHERE job IN ("ANALYST", "PROGRAMMER")  
-> AND sal >= 1000 AND sal <= 7000;
```

empno	ename	job	sal
7788	SCOTT	ANALYST	3000.00
7902	FORD	ANALYST	3000.00
7999	BATES	ANALYST	1300.00

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

```
mysql>
```



mysql>

mysql> SELECT \* FROM EMP;

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	SCOTT	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

15 rows in set (0.00 sec)

mysql> SELECT empno, ename, job, sal

-> FROM EMP

-> WHERE (job IN ("ANALYST", "PROGRAMMER"))

-> AND (sal >= 1300 OR sal+IFNULL(comm,0) >= 1800)

-> ORDER BY job, ename, sal;

empno	ename	job	sal
7999	BATES	ANALYST	1300.00
7902	FORD	ANALYST	3000.00
7788	SCOTT	ANALYST	3000.00

3 rows in set (0.00 sec)

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
SELECT empno, ename, job, sal  
FROM EMP  
WHERE job IN ("ANALYST", "PROGRAMMER")  
AND ename LIKE "%ES"  
AND sal BETWEEN 1000 AND 7000;
```

```
mysql> SELECT * FROM emp;
```

EMPNO	ENAME	JOB	SAL	DEPTNO
1	SMITH	CLERK	800	20
2	ALLEN	SALESMAN	1600	30
3	WARD	SALESMAN	1250	30
4	JONES	MANAGER	2975	20
5	MARTIN	SALESMAN	1250	30
6	BLAKE	MANAGER	2850	30
7	CLARK	MANAGER	2450	10
8	SCOTT	ANALYST	3000	20
9	KING	PRESIDENT	5000	10
10	TURNER	SALESMAN	1500	30
11	ADAMS	CLERK	1100	20
12	JAMES	CLERK	950	30
13	FORD	ANALYST	3000	20
14	MILLER	CLERK	1300	10
15	BATES	ANALYST	1300	NULL

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

```
mysql> SELECT empno, ename, job, sal
-> FROM EMP
-> WHERE job IN ("ANALYST", "PROGRAMMER")
-> AND ename LIKE "%ES"
-> AND sal BETWEEN 1000 AND 7000;
```

empno	ename	job	sal
15	BATES	ANALYST	1300

```
1 row in set (0.01 sec)
```

```
mysql> SELECT * FROM DEPT;
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

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

```
mysql> SELECT * FROM EMP;
```

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	SCOTT	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

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

```
SELECT empno, ename, job, sal, deptno
FROM EMP
WHERE sal > (SELECT MIN(sal)
              FROM EMP
              WHERE deptno IN (10, 20))

ORDER BY deptno, ename;
```

```

mysql>
mysql> SELECT empno, ename, job, sal, deptno
-> FROM EMP
-> WHERE sal > (SELECT MIN(sal)
->              FROM EMP
->              WHERE deptno IN (10, 20))
-> ORDER BY deptno, ename;

```

empno	ename	job	sal	deptno
7999	BATES	ANALYST	1300.00	NULL
7782	CLARK	MANAGER	2450.00	10
7839	KING	PRESIDENT	5000.00	10
7934	MILLER	CLERK	1300.00	10
7876	ADAMS	CLERK	1100.00	20
7902	FORD	ANALYST	3000.00	20
7566	JONES	MANAGER	2975.00	20
7788	SCOTT	ANALYST	3000.00	20
7499	ALLEN	SALESMAN	1600.00	30
7698	BLAKE	MANAGER	2850.00	30
7900	JAMES	CLERK	950.00	30
7654	MARTIN	SALESMAN	1250.00	30
7844	TURNER	SALESMAN	1500.00	30
7521	WARD	SALESMAN	1250.00	30

```

14 rows in set (0.01 sec)

```

# Διαγραφή πινάκων

/\* Δηλώσεις διαγραφής πινάκων \*/

DROP TABLE EMP;

DROP TABLE DEPT;



# Θυμηθείτε τις διαφορές στην υλοποίηση βάσεων δεδομένων, π.χ. Oracle, mySQL

Υπάρχουν διαφορές στην υλοποίηση ανάλογα με το αν χρησιμοποιούμε το προϊόν της Oracle ή της mySQL

# mySQL vs Oracle

mySQL	Oracle
CREATE DATABASE new_personnel;	
USE new_personnel;	
CREATE TABLE DEPT( DEPTNO INT(2) NOT NULL, DNAME VARCHAR(14), LOC VARCHAR(14));	CREATE TABLE DEPT( DEPTNO NUMBER(2) NOT NULL, DNAME VARCHAR2(14), LOC VARCHAR2(14));
CREATE TABLE EMP( EMPNO INT(4) NOT NULL, ENAME VARCHAR(10), JOB VARCHAR(25), HIREDATE DATE, MGR INT(4), SAL FLOAT(7,2), COMM FLOAT(7,2), DEPTNO INT(2));	CREATE TABLE EMP( EMPNO NUMBER(4) NOT NULL, ENAME VARCHAR2(10), JOB VARCHAR2(25), HIREDATE DATE, MGR NUMBER(4), SAL NUMBER(7,2), COMM NUMBER(7,2), DEPTNO NUMBER(2));
INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (10, 'ACCOUNTING', 'NEW YORK'); INSERT INTO EMP VALUES (10, 'CODD', 'ANALYST', '1989/01/01', 15, 3000, NULL, 10);	INSERT INTO DEPT(DEPTNO, DNAME, LOC) VALUES (10, 'ACCOUNTING', 'NEW YORK'); INSERT INTO EMP VALUES (10, 'CODD', 'ANALYST', '01/01/1989', 15, 3000, NULL, 10);
SELECT * FROM EMP; SELECT * FROM DEPT;	SELECT * FROM EMP; SELECT * FROM DEPT;
DROP TABLE EMP; DROP TABLE DEPT;	DROP TABLE EMP; DROP TABLE DEPT;
DROP DATABASE NEW_PERSONNEL;	
SHOW TABLES;	SELECT * FROM Tab;



**Τέλος Ενότητας**

Ερωτήσεις;

**Σημειώματα**

# Σημείωμα Αναφοράς

Copyright Πανεπιστήμιο Δυτικής Αττικής, Χ. Σκουρλάς 2021.

Χ. Σκουρλάς. «Βάσεις Δεδομένων Ι. Ενότητα 3: Εισαγωγή στην υλοποίηση σχεσιακών βάσεων δεδομένων με χρήση γλώσσας SQL (επεκτάσεις στην παρουσίαση της ενότητας 2)». Έκδοση: 1.0. Αθήνα 2021. Διαθέσιμο από τη δικτυακή διεύθυνση: [pyles.uniwa.gr](http://pyles.uniwa.gr).

# Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Μη Εμπορική Χρήση Παρόμοια Διανομή 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».



[1] <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Ως **Μη Εμπορική** ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου, για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό όφελος (π.χ. διαφημίσεις) από την προβολή του έργου σε διαδικτυακό τόπο

Ο δικαιούχος μπορεί να παρέχει στον αδειοδόχο ξεχωριστή άδεια να χρησιμοποιεί το έργο για εμπορική χρήση, εφόσον αυτό του ζητηθεί.

# Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.