

# Το Σχεσιακό μοντέλο και η γλώσσα SQL

Μανόλης Γεργατσούλης (manolis@ionio.gr)

Αναπληρωτής Καθηγητής

Ομάδα Βάσεων Δεδομένων και Πληροφοριακών Συστημάτων,  
Τμήμα Αρχιονομίας – Βιβλιοθηκονομίας, Ιόνιο Πανεπιστήμιο

# Γιατί Μελετάμε το Σχεσιακό Μοντέλο;

- Είναι το πιο διαδεδομένο μοντέλο.
  - Σχεσιακά DBMS: DB2 της IBM, Oracle, Informix, MS-ACCESS και MS-SQLServer της Microsoft, Sybase, κ.λ.π.
- Τελευταίος ανταγωνιστής: αντικειμενοστραφές μοντέλο.
  - ObjectStore, Versant, Ontos
  - Πρόσφατη σύνθεση των δύο μοντέλων: *αντικείμενο-σχεσιακό μοντέλο*
    - Informix Universal Server, UniSQL, O2, Oracle, DB2

# Σχεσιακές Βάσεις Δεδομένων: Ορισμοί

- **Σχεσιακή ΒΔ**: ένα σύνολο **σχέσεων (πινάκων)**
- **Σχέση**: αποτελείται από 2 συστατικά:
  - **Στιγμιότυπο** : ένας **πίνακας**, με γραμμές και στήλες.  
#Γραμμές = πληθυσμός, #πεδία = βαθμός.
  - **Σχήμα** : προσδιορίζει το όνομα της σχέσης, και τα ονόματα και τους τύπους κάθε στήλης.
    - Π.χ. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gra*: real).
- Μια σχέση μπορεί να θεωρηθεί ως **σύνολο γραμμών** ή **πλειάδων** (οι γραμμές είναι διαφορετικές μεταξύ τους).

# Παράδειγμα: Στιγμιότυπο της Σχέσης Students

Πεδία, γνωρίσματα, στήλες

Ονόματα πεδίων

Πλειάδες (εγγραφές,  
γραμμές)

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Πληθυσμός = 3, βαθμός = 5, διακριτές γραμμές

Πρέπει όλες οι στήλες σε ένα στιγμιότυπο μιας  
σχέσης να είναι διακριτές

# Σχεσιακές Γλώσσες Αιτημάτων

- Ισχυρό πλεονέκτημα του σχεσιακού μοντέλου: υποστηρίζει την υποβολή απλών αλλά ισχυρών **αιτημάτων** πάνω στα δεδομένα.
- Τα αιτήματα μπορούν να διατυπωθούν διαισθητικά και το DBMS είναι υπεύθυνο για τον αποδοτικό υπολογισμό τους.
  - Σημείο κλειδί: η ακριβής σημασιολογία των σχεσιακών αιτημάτων.
  - Επιτρέπει στο βελτιστοποιητή την εκτενή αναδιάταξη των λειτουργιών, εξασφαλίζοντας παράλληλα ότι η απάντηση δεν αλλάζει.

# Η Γλώσσα Αιτημάτων SQL

- Αναπτύχθηκε από την IBM (system R) τη δεκαετία του 1970.
- Απαίτηση για πρότυπα αφού χρησιμοποιείται από πολλούς κατασκευαστές λογισμικού.
- Πρότυπα:
  - SQL-86
  - SQL-89 (μικρές βελτιώσεις)
  - SQL-92 (σημαντικές βελτιώσεις, το τρέχον πρότυπο)
  - SQL-99 (ουσιαστικές επεκτάσεις).

# Η Γλώσσα Αιτημάτων SQL

- Για να βρούμε όλους τους φοιτητές ηλικίας 18 ετών γράφουμε:

Αποτέλεσμα

```
SELECT *  
FROM Students S  
WHERE S.age=18
```

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@ee	18	3.2

Για να πάρουμε μόνο τα ονόματα και τα *logins*, αντικαθιστούμε την πρώτη γραμμή με:

```
SELECT S.name, S.login
```

# Αιτήματα σε Πολλαπλές Σχέσεις

- Τι υπολογίζει η ακόλουθη ερώτηση;

```
SELECT S.name, E.cid  
FROM Students S, Enrolled E  
WHERE S.sid=E.sid AND E.grade= 'A'
```

Όταν δίνεται το ακόλουθο στιγμιότυπο της *Enrolled* (είναι δυνατό όταν το DBMS εξασφαλίζει αναφορική ακεραιότητα;):

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enrolled

sid	cid	grade
53831	Carnatic101	C
53831	Reggae203	B
53650	Topology112	A
53666	History105	B

αποτέλεσμα:

S.name	E.cid
Smith	Topology112

# Δημιουργία Σχέσεων με την SQL

- Δημιουργεί τη σχέση *Students*. Παρατηρήστε ότι προσδιορίζεται και ο τύπος (**domain**) κάθε πεδίου, και επιβάλλεται από το DBMS κάθε φορά που προστίθενται ή τροποποιούνται πλειάδες.
- Άλλο παράδειγμα: ο πίνακας *Enrolled* διατηρεί πληροφορίες σχετικά με τα μαθήματα που παίρνουν οι φοιτητές.

```
CREATE TABLE Students  
  (sid CHAR(20),  
   name CHAR(20),  
   login CHAR(10),  
   age INTEGER,  
   gpa REAL)
```

```
CREATE TABLE Enrolled  
  (sid CHAR(20),  
   cid CHAR(20),  
   grade CHAR(2))
```

# Ακύρωση και Μεταβολή Σχέσεων

DROP TABLE Students

- Ακυρώνει τη σχέση *Students*. Η πληροφορία του σχήματος και οι πλειάδες διαγράφονται.

ALTER TABLE Students

ADD COLUMN firstYear integer

- ❖ Το σχήμα της σχέσης *Students* μεταβάλλεται και προστίθεται ένα νέο πεδίο. Κάθε πλειάδα στο τρέχον στιγμιότυπο επεκτείνεται με τη τιμή ***null*** στο νέο πεδίο.

# Εισαγωγή και Διαγραφή Πλειάδων

- Μπορούμε να εισάγουμε μια πλειάδα ως εξής:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

Μπορούμε να διαγράψουμε όλες τις πλειάδες που ικανοποιούν μια συνθήκη (π.χ., *name = 'Smith'*):

```
DELETE
FROM Students S
WHERE S.name = 'Smith'
```

# Περιορισμοί Ακεραιότητας (ICs)

- IC: συνθήκη που πρέπει να αληθεύει *για κάθε* στιγμιότυπο της ΒΔ (π.χ. περιορισμοί πεδίου ορισμού).
  - Τα ICs προσδιορίζονται όταν ορίζεται το σχήμα.
  - Τα ICs ελέγχονται όταν τροποποιούνται οι σχέσεις.
- *Έγκυρο* στιγμιότυπο μιας σχέσης είναι εκείνο που ικανοποιεί όλα τα ICs.
  - Τα DBMS δεν επιτρέπουν μη έγκυρα στιγμιότυπα.
- Αν ένα DBMS ελέγχει τα ICs, τα αποθηκευμένα δεδομένα είναι πιστότερα στη σημασία που έχουν στον πραγματικό κόσμο.
  - Επίσης αποφεύγονται λάθη στην εισαγωγή στοιχείων!

# Περιορισμοί Κύριου Κλειδιού

- Ένα σύνολο πεδίων είναι **κλειδί** για μια σχέση αν:
  1. Δεν επιτρέπεται σε δύο διαφορετικές πλειάδες να έχουν τις ίδιες τιμές σε όλα τα πεδία του κλειδιού, και
  2. Το προηγούμενο δεν ισχύει για κανένα υποσύνολο του κλειδιού.
    - Αν το 2 δεν ισχύει τότε έχουμε ένα **εμπλουτισμένο κλειδί**.
    - Αν υπάρχουν περισσότερα από 1 **υποψήφια κλειδιά** για μια σχέση, τότε επιλέγεται (από τον DBA) ένα από αυτά να είναι το **κύριο κλειδί**.
- Π.χ., το *sid* είναι κλειδί για τη *Students*. (Τι λέτε για το *name*;) Το σύνολο {*sid*, *gra*} είναι ένα εμπλουτισμένο κλειδί.

# Κύρια Κλειδιά στην SQL

- Η δήλωση του κλειδιού ενός πίνακα γίνεται με την **PRIMARY KEY** ενώ ακολουθεί το όνομα ή τα ονόματα των πεδίων που θα αποτελέσουν το κλειδί σε παρένθεση.

```
CREATE TABLE Students  
  (sid CHAR(20),  
   name CHAR(20),  
   login CHAR(10),  
   age INTEGER,  
   gpa REAL,  
   PRIMARY KEY (sid) )
```

# Ξένα Κλειδιά, Αναφορική Ακεραιότητα

- Ξένο κλειδί : Σύνολο πεδίων μιας σχέσης το οποίο χρησιμοποιείται για να `αναφέρεται` σε μια πλειάδα άλλης σχέσης σαν `λογικός δείκτης` (αντιστοιχεί στο κύριο κλειδί μιας άλλης σχέσης).
- Π.χ., το *sid* είναι ξένο κλειδί στη σχέση *Enrolled* αναφερόμενο στη σχέση *Students*:
  - *Enrolled(sid: string, cid: string, grade: string)*
  - Με την επιβολή όλων των περιορισμών ξένου κλειδιού πετυχαίνουμε αναφορική ακεραιότητα, δηλ. αποφεύγονται αναφορές σε ανύπαρκτες πλειάδες.
  - Μοντέλο δεδομένων που δεν έχει αναφορική ακεραιότητα:
    - Σύνδεσμοι στην HTML!

# Ξένα Κλειδιά στην SQL

- Μόνο οι φοιτητές που έχουν καταχωρηθεί στη σχέση *Students* επιτρέπεται να εγγράφονται σε μαθήματα.

```
CREATE TABLE Enrolled  
  (sid CHAR(20), cid CHAR(20), grade CHAR(2),  
   PRIMARY KEY (sid,cid),  
   FOREIGN KEY (sid) REFERENCES Students )
```

## Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

## Students

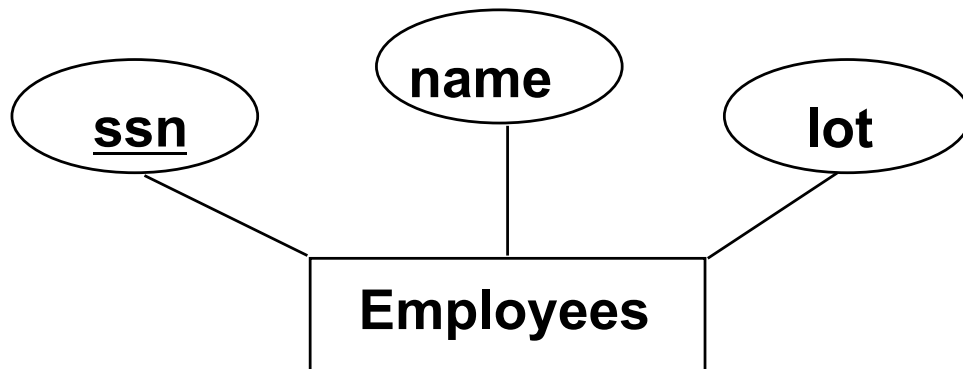
sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

# Επιβολή Αναφορικής Ακεραιότητας

- Έστω οι σχέσεις *Students* και *Enrolled*. Το *sid* της *Enrolled* είναι ξένο κλειδί και αναφέρεται στη *Students*.
- Τι πρέπει να γίνει αν γίνει προσπάθεια να εισαχθεί πλειάδα της *Enrolled* με ανύπαρκτο κωδικό φοιτητή (*id*); (Να απορριφθεί!)
- Τι πρέπει να γίνεται κατά τη διαγραφή πλειάδας της *Students*; Δυνατές επιλογές:
  - Να διαγραφούν όλες οι πλειάδες της *Enrolled* που αναφέρονται σ' αυτήν.
  - Να απαγορευτεί η διαγραφή της αναφερόμενης πλειάδας της *Students*.
  - Να τοποθετηθεί μια προαποφασισμένη τιμή στο *sid* όλων των πλειάδων της *Enrolled* που αναφέρονται σ' αυτήν.
  - Να τοποθετηθεί η ειδική τιμή *null*, που δηλώνει 'άγνωστο' ή 'μη εφαρμόσιμο', στο *sid* των πλειάδων της *Enrolled* που αναφέρονται σ' αυτήν. Αυτό δεν επιτρέπεται εδώ γιατί το *sid* συμμετέχει στο κύριο κλειδί της *Enrolled*.
- Παρόμοια για την ενημέρωση του κύριου κλειδιού της *Students*.

# Λογικός Σχεδιασμός Βάσης Δεδομένων: Από το ER στο Σχεσιακό Σχήμα

- Από τα σύνολα οντοτήτων στους πίνακες.

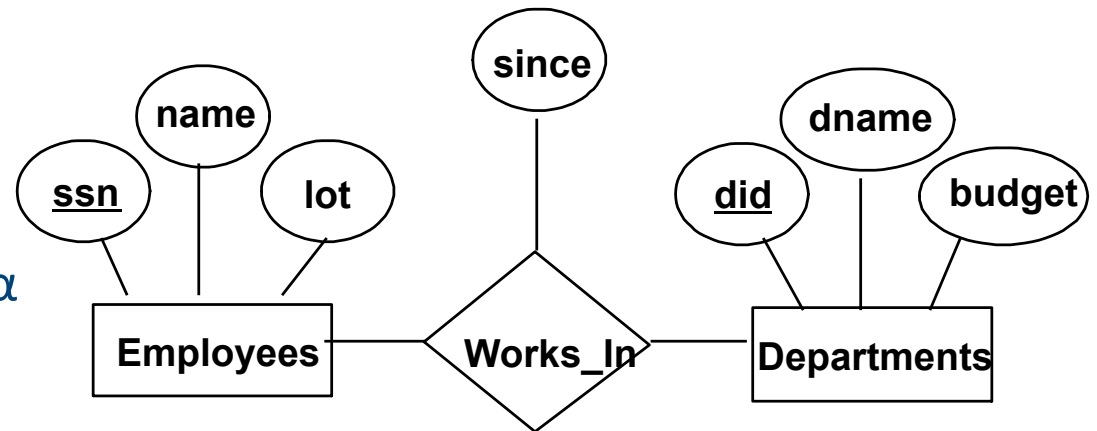


- Μια οντότητα μετατρέπεται σε έναν πίνακα με στήλες τα γνωρίσματα της οντότητας:

```
CREATE TABLE Employees  
(ssn CHAR(11),  
name CHAR(20),  
lot INTEGER,  
PRIMARY KEY (ssn))
```

# Από τις Συσχετίσεις στους Πίνακες

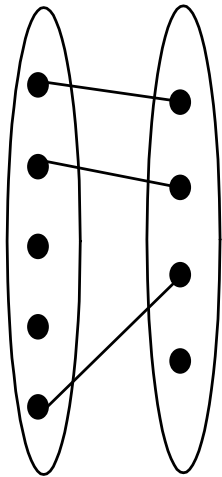
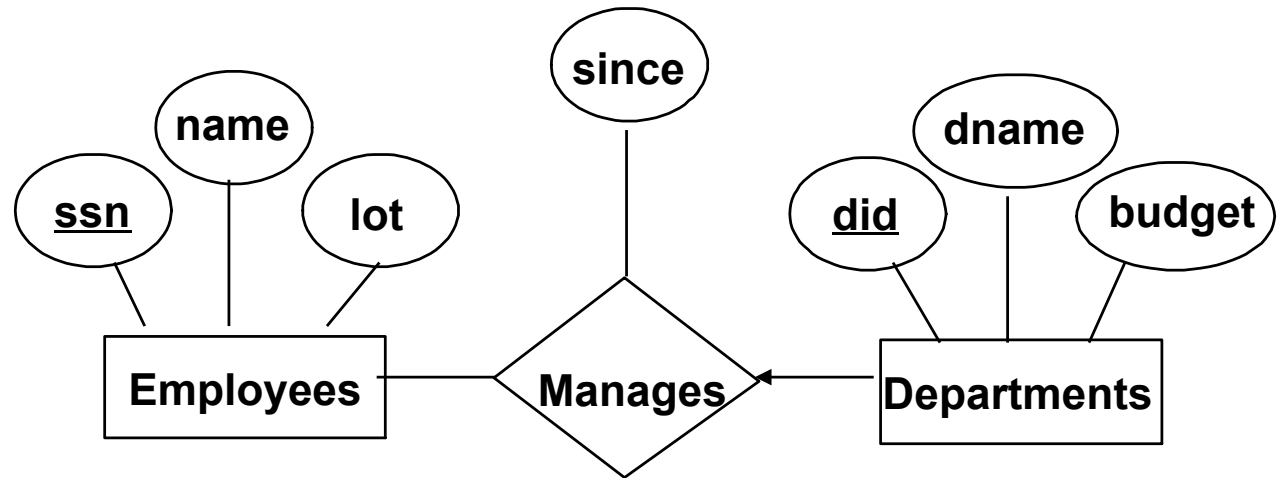
- Κατά τη μετατροπή συνόλου συσχετίσεων σε σχέση, τα γνωρίσματα της σχέσης πρέπει να περιλαμβάνουν:
  - Κλειδιά για κάθε σύνολο οντοτήτων που συμμετέχει (σαν ξένα κλειδιά).
    - Αυτό το σύνολο γνωρισμάτων αποτελεί ένα **εμπλουτισμένο κλειδί** για τη σχέση.
  - Όλα τα περιγραφικά γνωρίσματα της συσχέτισης.



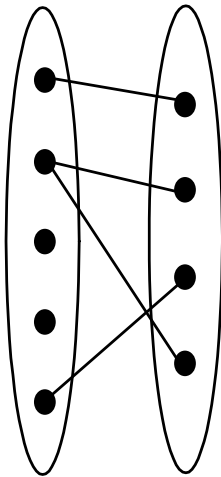
```
CREATE TABLE Works_In(  
  ssn CHAR(11),  
  did INTEGER,  
  since DATE,  
  PRIMARY KEY (ssn, did),  
  FOREIGN KEY (ssn)  
    REFERENCES Employees,  
  FOREIGN KEY (did)  
    REFERENCES Departments)
```

# Επισκόπηση: Περιορισμοί Κλειδιού

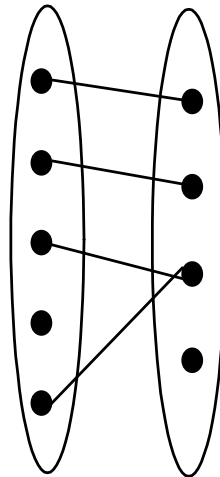
- Κάθε τμήμα έχει το πολύ ένα διευθυντή, με βάση τον περιορισμό κλειδιού στη σχέση Manages.



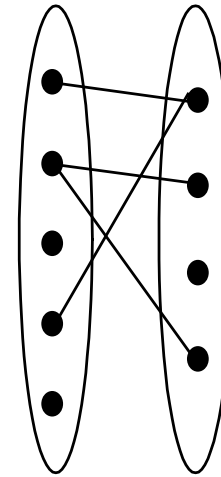
1-προς-1



1-προς Πολλά



Πολλά-προς-1

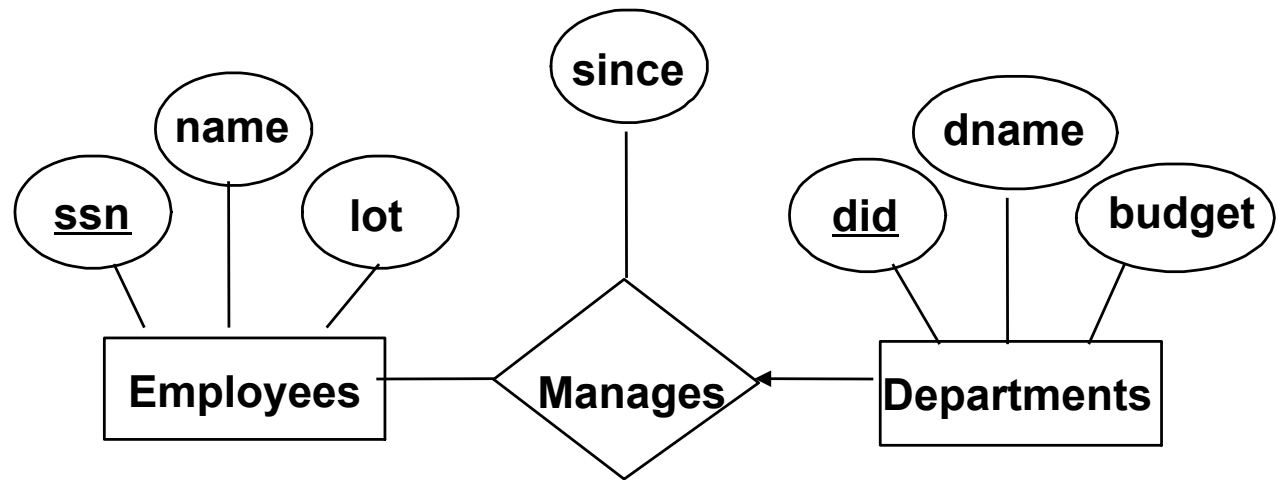


Πολλά-προς-Πολλά

Πως μετατρέπονται  
στο σχεσιακό  
μοντέλο;

# Μετατρέποντας ER διαγράμματα με Περιορισμούς Κλειδιού (1/2)

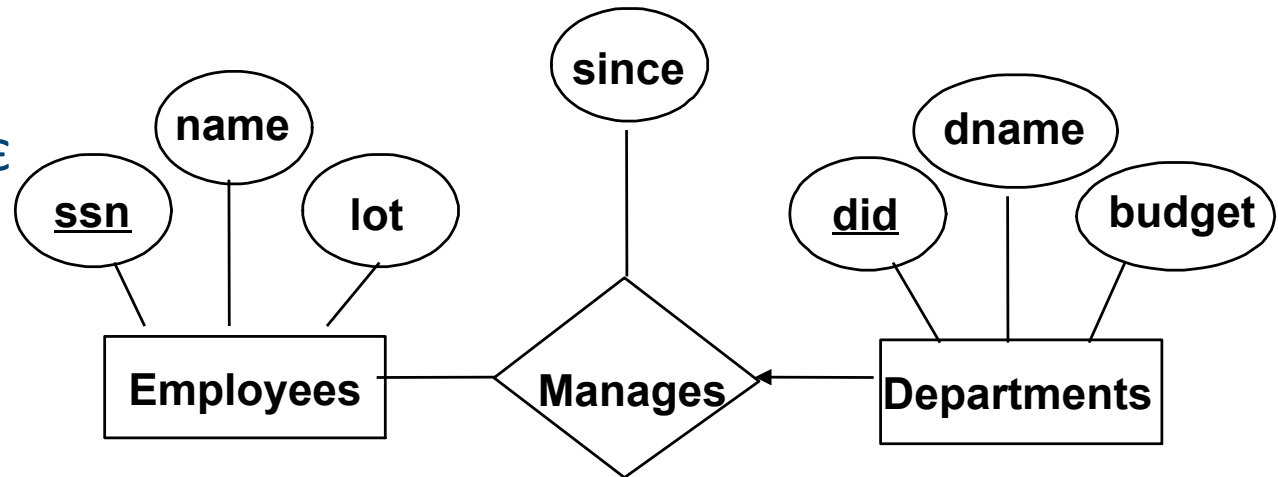
- 1ος τρόπος (όπως και πριν):
  - Η συσχέτιση απεικονίζεται σαν πίνακας:
  - Ξεχωριστοί πίνακες *Employees* και *Departments*.
  - Προσοχή: το *did* είναι τώρα το κλειδί!



```
CREATE TABLE Manages(  
  ssn CHAR(11),  
  did INTEGER,  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  FOREIGN KEY (did) REFERENCES Departments)
```

# Μετατρέποντας ER διαγράμματα με Περιορισμούς Κλειδιού (2/2)

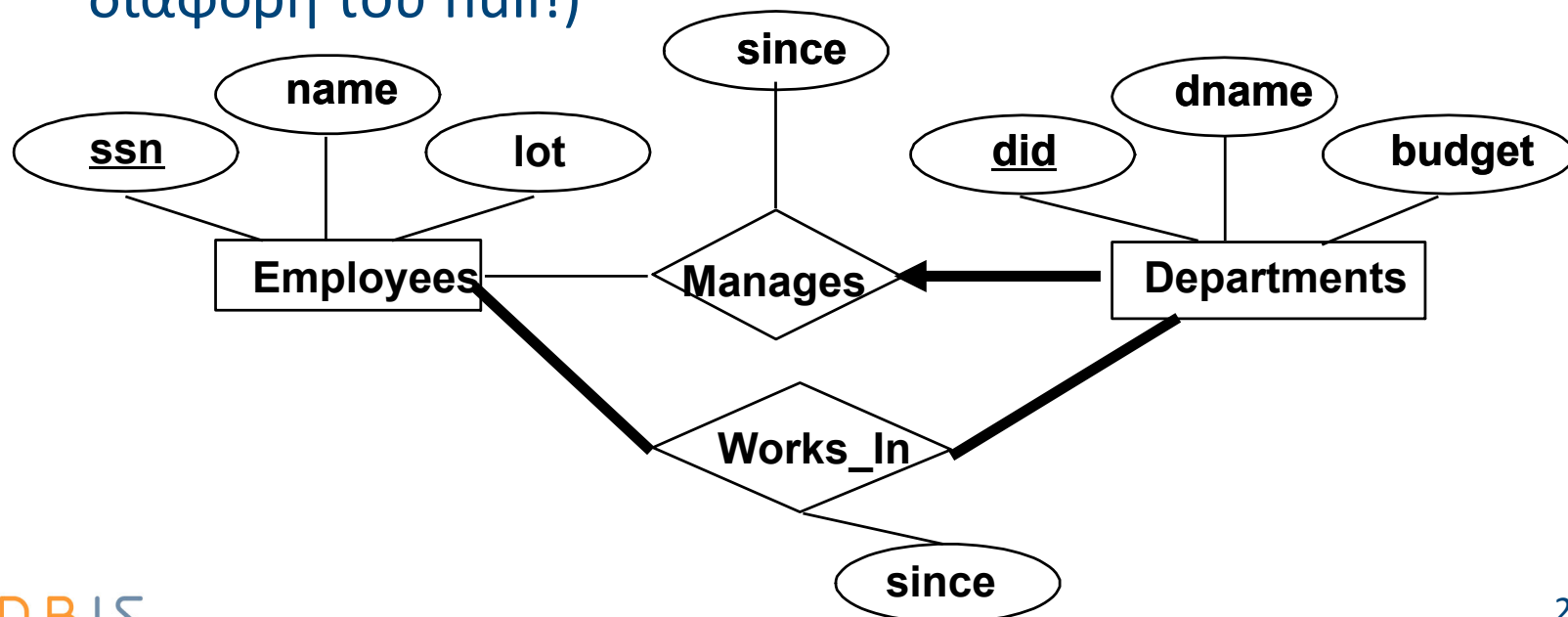
- 2ος τρόπος Αφού κάθε τμήμα έχει έναν μοναδικό διευθυντή (το πολύ), θα μπορούσαμε εναλλακτικά να συνδυάσουμε τους *Manages* και *Departments* δημιουργώντας έναν ενιαίο πίνακα.



```
CREATE TABLE Dept_Mgr(  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  ssn CHAR(11),  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees)
```

# Επισκόπηση: Περιορισμοί Συμμετοχής

- Έχει κάθε τμήμα οπωσδήποτε ένα διευθυντή;
  - Αν ναι, τότε αυτό αποτελεί περιορισμό συμμετοχής: η συμμετοχή του πίνακα *Departments* στο *Manages* ονομάζεται ολική (σε αντίθεση με τη μερική).
  - Κάθε τιμή του *did* στον πίνακα *Departments* πρέπει να εμφανίζεται σε μια γραμμή του *Manages* (με τιμή του *ssn* διάφορη του null!)



# Περιορισμοί Συμμετοχής στην SQL

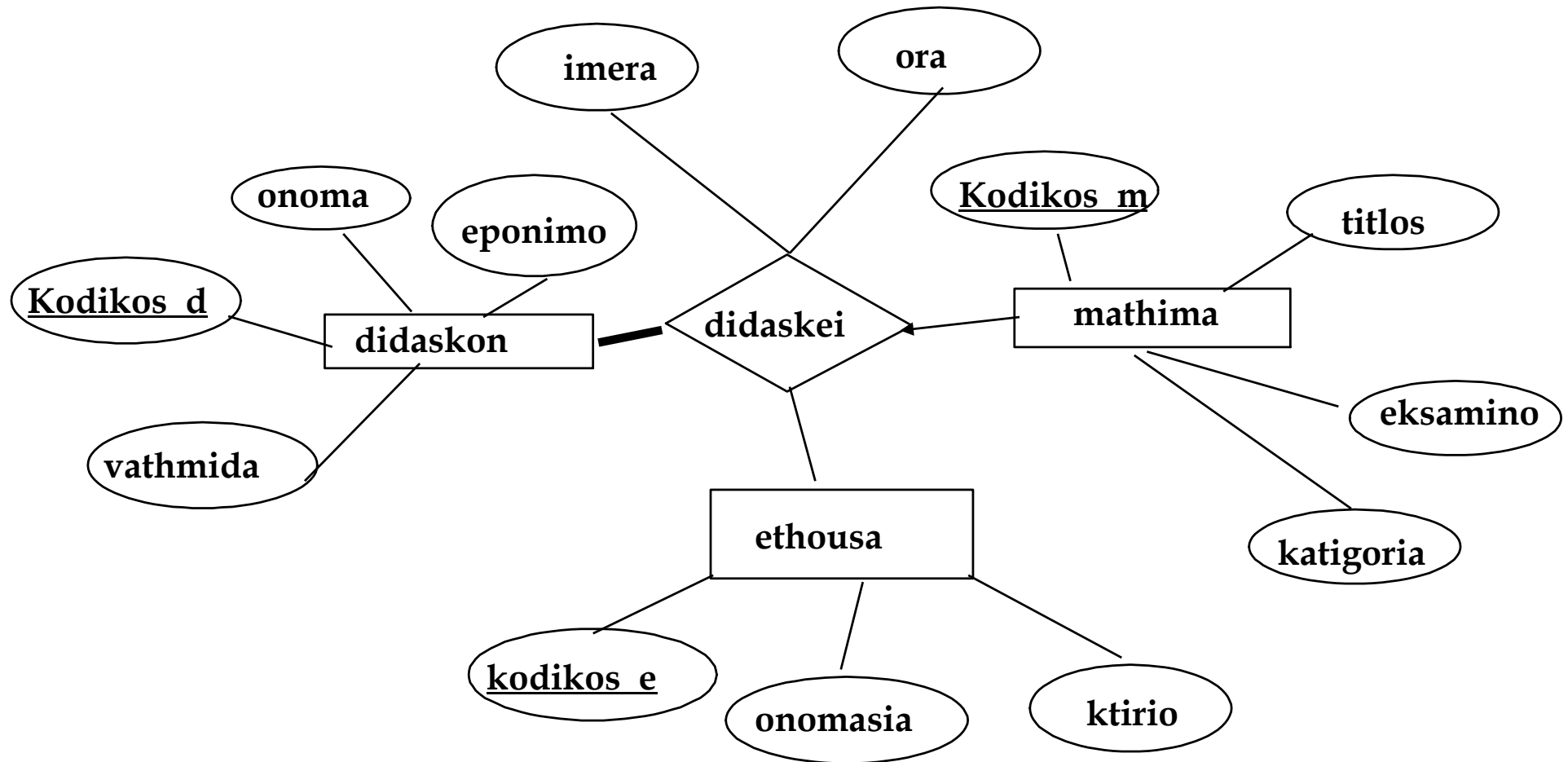
- Μπορούμε να ορίσουμε περιορισμούς συμμετοχής που περιλαμβάνουν ένα σύνολο οντοτήτων σε μια δυαδική συσχέτιση, όχι όμως πολλά πέρα από αυτά (χωρίς προσφυγή σε CHECK περιορισμούς).

```
CREATE TABLE Dept_Mgr(  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  ssn CHAR(11) NOT NULL,  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  ON DELETE NO ACTION)
```

# ΕΚΦΩΝΗΣΗ ΑΣΚΗΣΗΣ

- Ζητείται να σχεδιαστεί βάση δεδομένων με στόχο τη διαχείριση των αναθέσεων μαθημάτων του τμήματος Αρχειονομίας & Βιβλιοθηκονομίας και των αιθουσών για τη διεξαγωγή των μαθημάτων αυτών. Για τη σχεδίαση της βάσης θεωρείστε ότι ισχύουν οι ακόλουθες παραδοχές:
  - Η διδασκαλία κάθε μαθήματος ανατίθεται σε έναν μόνο καθηγητή.
  - Κάθε καθηγητής διδάσκει τουλάχιστον ένα μάθημα.

# ER-ΔΙΑΓΡΑΜΜΑ



# ΟΡΙΣΜΟΣ ΠΙΝΑΚΩΝ ΣΕ SQL

```
CREATE TABLE didaskon  
  ( kodikos_d CHAR(5),  
    onoma      CHAR(15),  
    eponimo    CHAR(20),  
    vathmida   CHAR(2),  
    PRIMARY KEY (kodikos_d));
```

```
CREATE TABLE mathima  
  ( kodikos_m CHAR(5),  
    titlos     CHAR(15),  
    eksamino  CHAR(2),  
    katigoria CHAR(2),  
    PRIMARY KEY (kodikos_m));
```

```
CREATE TABLE ethousa  
  ( kodikos_e CHAR(5),  
    onomasia  CHAR(15),  
    ktirio    CHAR(2),  
    PRIMARY KEY (kodikos_e));
```

# ΟΡΙΣΜΟΣ ΠΙΝΑΚΩΝ ΣΕ SQL

## (ΣΥΝΕΧΕΙΑ)

```
CREATE TABLE didaskei
( kodikos_d CHAR(5),
  kodikos_m CHAR(5),
  kodikos_e CHAR(5),
  imera CHAR(10),
  ora CHAR(10),
  PRIMARY KEY (kodikos_m),
  FOREIGN KEY (kodikos_m) REFERENCES mathima,
  FOREIGN KEY (kodikos_d) REFERENCES didaskon,
  FOREIGN KEY (kodikos_e) REFERENCES ethousa
);
```

## ΑΙΤΗΜΑ 1:

Να βρεθούν τα ονόματα και τα επώνυμα όλων των διδασκόντων.

SELECT ονομα, εponimo FROM didaskon	SELECT D.ονομα, D.εponimo FROM didaskon D
1 <sup>ος</sup> τρόπος	2 <sup>ος</sup> τρόπος

## ΑΙΤΗΜΑ 2:

Να βρεθούν οι τίτλοι όλων των μαθημάτων του 'Δ' εξαμήνου.

<pre>SELECT titlos FROM mathima WHERE eksamino = 'Δ'</pre>	<pre>SELECT M.titlos FROM mathima M WHERE M.eksamino = 'Δ'</pre>
1 <sup>ος</sup> τρόπος	2 <sup>ος</sup> τρόπος

## ΑΙΤΗΜΑ 3:

Ζητείται πίνακας ο οποίος περιλαμβάνει τα ονόματα και τα επώνυμα των διδασκόντων και τα μαθήματα που διδάσκουν.

```
SELECT D1.onoma, D1.eponimo, M.titlos  
FROM didaskon D1, didaskei D2, mathima M  
WHERE D1.kodikos_d = D2.kodikos_d AND  
      D2.kodikos_m = M.kodikos_m
```

## ΑΙΤΗΜΑ 4:

Ζητείται πίνακας ο οποίος περιλαμβάνει τους τίτλους των μαθημάτων που διδάσκονται στην αίθουσα με κωδικό 'A1'.

```
SELECT M.titlos
FROM didaskei D, mathima M
WHERE D.kodikos_e = 'A1' AND
      D.kodikos_m = M.kodikos_m
```

## ΑΙΤΗΜΑ 5:

Ζητείται πίνακας με τους τίτλους των μαθημάτων που διδάσκει ο διδάσκων με επώνυμο 'ΓΕΡΓΑΤΣΟΥΛΗΣ' συνοδευόμενους από το όνομα των αιθουσών στις οποίες διδάσκονται τα μαθήματα αυτά.

```
SELECT M.titlos, E.onomasia
FROM didaskei D1, mathima M, ethousa E, didaskon D2
WHERE D2.eponimo = 'ΓΕΡΓΑΤΣΟΥΛΗΣ' AND
      D2.kodikos_d = D1.kodikos_d AND
      D1.kodikos_m = M.kodikos_m AND
      D1.kodikos_e = E.kodikos_e
```

## AΙΤΗΜΑ 6:

Να βρεθεί το πλήθος των διδασκόντων που ανήκουν στην βαθμίδα 'B'.

```
SELECT COUNT(*)  
FROM didaskon D  
WHERE D.vathmida = 'B'
```

## ΑΙΤΗΜΑ 7:

Να βρεθεί το πλήθος των μαθημάτων του 'Γ' εξαμήνου τα οποία διδάσκει ο διδάσκων με επώνυμο 'ΚΑΠΙΔΑΚΗΣ'.

```
SELECT COUNT(*)  
FROM didaskon D1, didaskei D2, mathima M  
WHERE D1.eponimo = 'ΚΑΠΙΔΑΚΗΣ' AND  
      D1.kodikos_d = D2.kodikos_d AND  
      D2.kodikos_m = M.kodikos_m AND  
      M.eksamino = 'Γ'
```

## AITHMA 8:

Να βρεθεί το πλήθος των μαθημάτων του 'Γ' εξαμήνου που διδάσκονται την 'ΤΕΤΑΡΤΗ'.

```
SELECT COUNT(*)  
FROM didaskei D, mathima M  
WHERE D.imeras = 'ΤΕΤΑΡΤΗ' AND  
       D.kodikos_m = M.kodikos_m AND  
       M.eksamino = 'Γ'
```