The Relational Data Model

Relational Data Model

- Προτάθηκε το 1970 από τον Codd
- Ισχυρό μαθηματικό υπόβαθρο
- Πρωτότυπα ξεκίνησαν να αναπτύσσονται στα τέλη της δεκαετίας του 1970
  - System R, IBM San Jose (Almaden Center)
  - Ingress, UC Berkeley
  - Micro DBMS U. Toronto

Relations, Domains, Cartesian Products in RDM

Σχέση (Relation):
Δεδομένων D_1, D_2, … D_n (**domains-πεδία ορισμού**),
μία **σχέση (relation)** στο σετ των n πεδίων ορισμού
είναι ένα **σετ από ταξινομημένα n-tuples** $< d_1, d_2, \ldots d_n >$
έτσι ώστε $d_1 \in D_1$, $d_2 \in D_2$, …$d_n \in D_n$.

D_1, … D_n είναι τα <u>πεδία ορισμού (domains)</u> της R

Ισοδύναμος ορισμός:

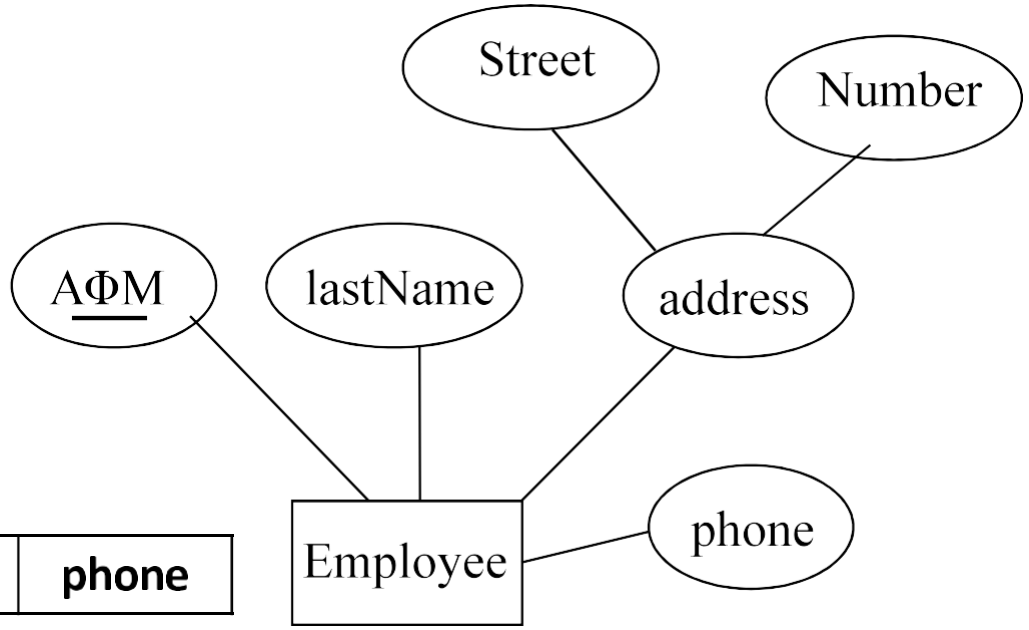
- **<u>Καρτεσιανό γινόμενο (Cartesian product)</u>**:

$D_1 \times D_2 \times \ldots \times D_n$ των D_1, … D_n είναι το σετ **όλων των δυνατών ordered n-tuples** $< d_1, \ldots d_n >$ έτσι ώστε $d_1 \in D_1$, …$d_n \in D_n$

Μία **<u>σχέση R</u>** στο D_1, … D_n είναι ένα **υποσύνολο του $D_1 \times \ldots \times D_n$**

Μετατροπή ER σε Σχεσιακό Σχήμα – Ισχυροί Τύποι Οντοτήτων

- Μετατροπή ισχυρών Τύπων Οντοτήτων σε Σχέσεις (Πίνακες)
  - Ένας πίνακας για κάθε ισχυρό τύπο οντοτήτων
  - Κάθε γνώρισμα γίνεται 1 στήλη στον πίνακα
  - Για τα σύνθετα γνωρίσματα αποθηκεύουμε τα απλά συστατικά τους

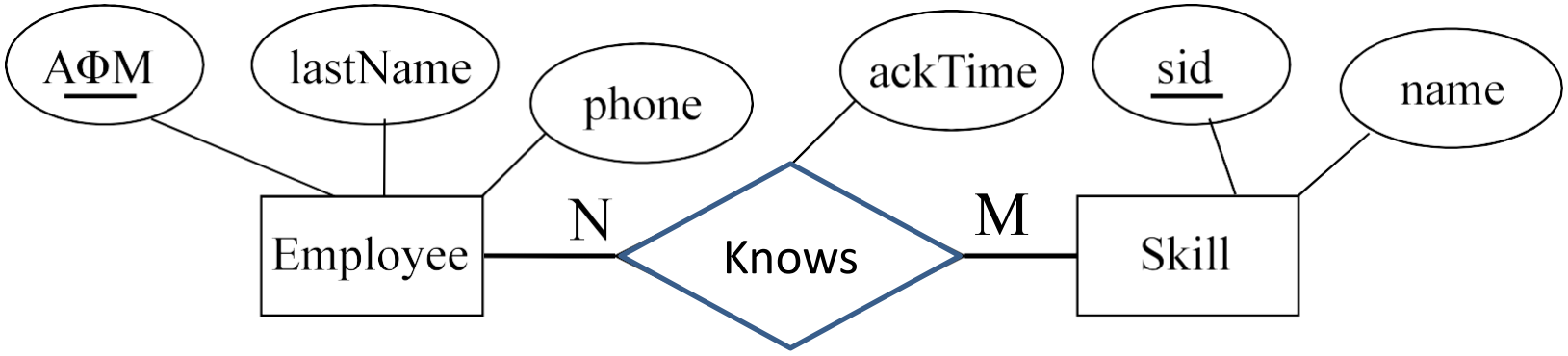


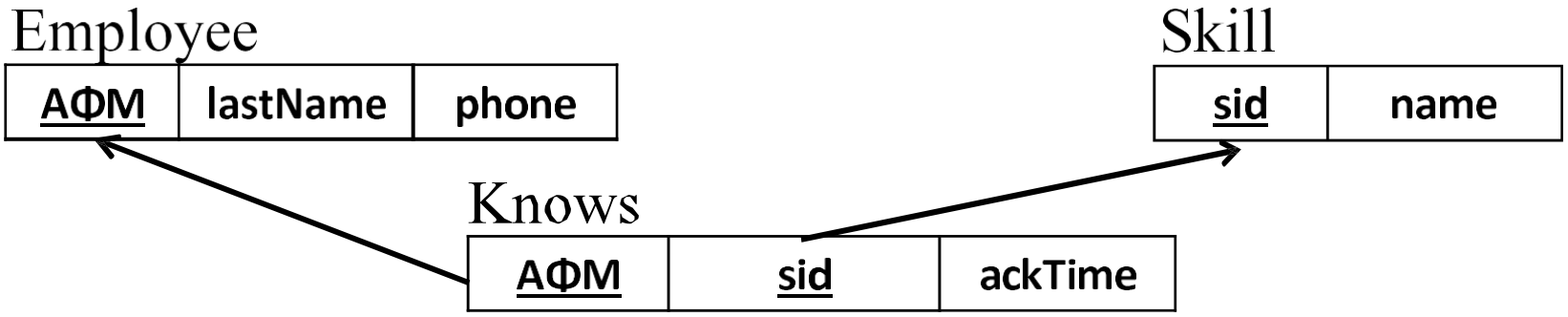
Employee

ΑΦΜ	lastName	Street	Number	phone

Μετατροπή ER σε Σχεσιακό Σχήμα – Συσχετίσεις Μ προς Ν

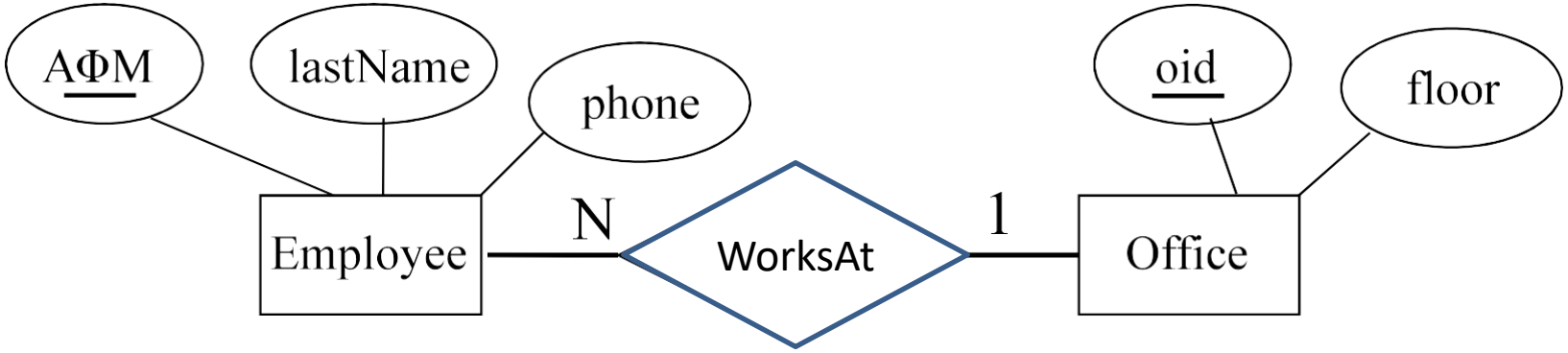
- Για κάθε τύπο συσχέτισης Μ-Ν δημιουργούμε μία νέα σχέση (πίνακα)
  - Γνωρίσματα: Τα κλειδιά των τύπων οντοτήτων που συσχετίζει
    - Μπορεί να χρειαστεί κάποιο να μετονομαστεί…
  - Περιέχει και τα γνωρίσματα της συσχέτισης (αν υπάρχουν)
  - Τα βελάκια υποδηλώνουν περιορισμούς ξένου κλειδιού (Foreign Key Constraints)





Μετατροπή ER σε Σχεσιακό Σχήμα – Συσχετίσεις 1 προς Ν

- Για κάθε τύπο συσχέτισης 1-Ν ΔΕ δημιουργούμε μία νέα σχέση (πίνακα)
  - Δεν είναι απόλυτο αν η πλευρά του Ν δεν έχει total participation, πολύ συνηθισμένο όμως και τότε
  - Στη σχέση στην πλευρά του Ν προσθέτουμε το κλειδί της άλλης πλευράς
  - Προσθέτουμε και τα γνωρίσματα της συσχέτισης (αν υπάρχουν)



Employee

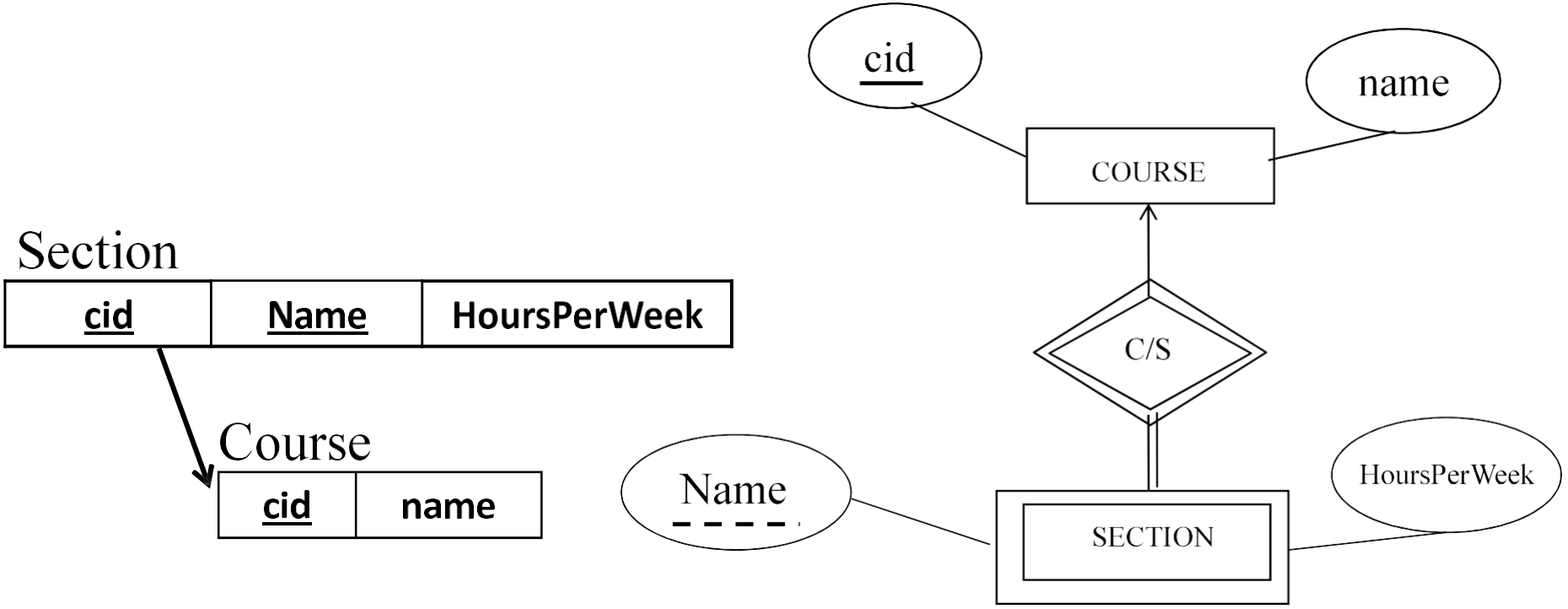
ΑΦΜ	**lastName**	**phone**	**office**

Office

oid	**floor**

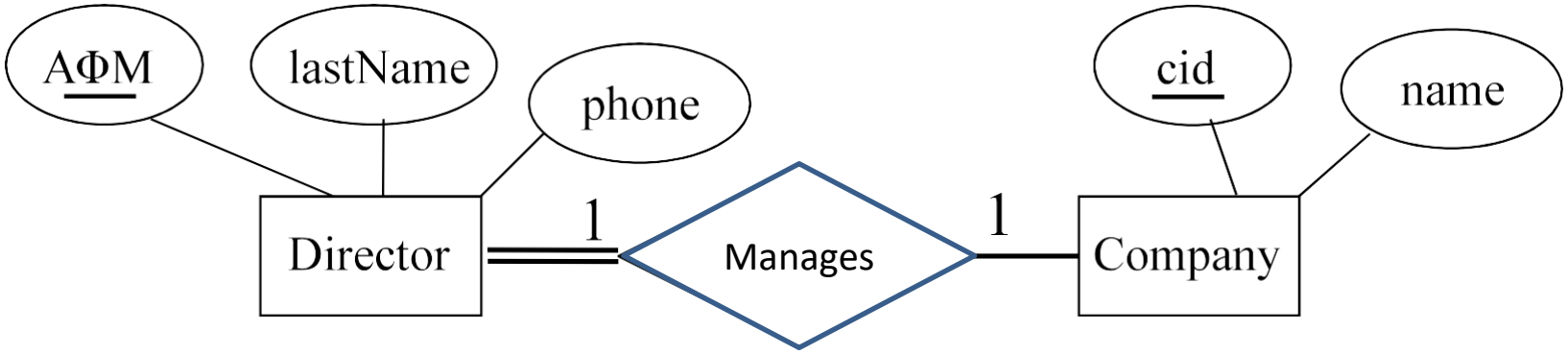
Μετατροπή ER σε Σχεσιακό Σχήμα – Ασθενείς Τύποι Οντοτήτων

- Για κάθε Ασθενή Τύπο Οντότητας δημιουργούμε μία νέα σχέση (πίνακα)
  - Στη σχέση προσθέτουμε το κλειδί της ισχυρής προσδιορίζουσας οντότητας
  - Πρωτεύον κλειδί: κλειδί προσδιορίζουσας οντότητας + μερικό κλειδί (αν υπάρχει)



Μετατροπή ER σε Σχεσιακό Σχήμα – Συσχετίσεις 1 προς 1

- Για κάθε τύπο συσχέτισης 1-1 ΔΕ δημιουργούμε μία νέα σχέση (πίνακα)
  - Στη σχέση Α προσθέτουμε το κλειδί της άλλης πλευράς Β
    - Καλύτερα να επιλέξουμε ως Α σχέση με total participation
  - Προσθέτουμε και τα γνωρίσματα της συσχέτισης (αν υπάρχουν)



Director

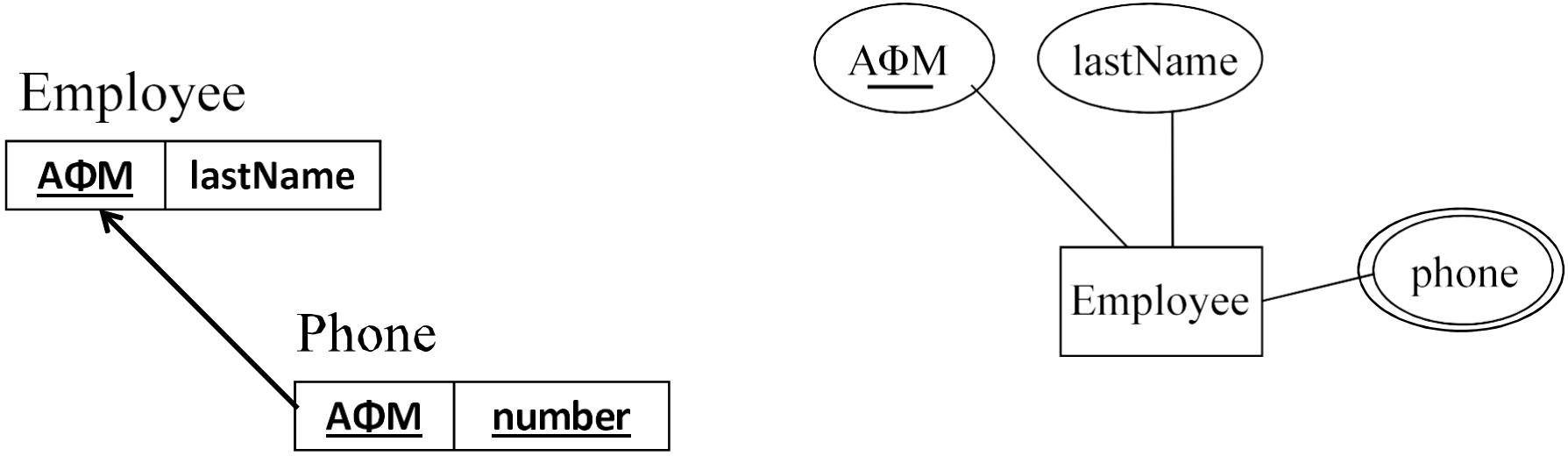
ΑΦΜ	lastName	phone	comp

Company

cid	name

Μετατροπή ER σε Σχεσιακό Σχήμα – Υπόλοιπες Περιπτώσεις

- Υπολογιζόμενα Γνωρίσματα δεν απεικονίζονται
- Πλειότιμα Γνωρίσματα: Μετατροπή τους σε επιπλέον Σχέση (Πίνακας)
  - Κλειδί τους: Κλειδί τύπου οντότητας + γνώρισμα
- Θα δούμε ISA στο τέλος



Περιορισμοί (Constraints) σε Σχεσιακό Μοντέλο Δεδομένων (RDM)

- **Candidate key:** μοναδικό προσδιοριστικό των πλειάδων
- **Primary key:** ένα επιλεγμένο candidate key
- Το σύστημα εγγυάται τη μοναδικότητα των τιμών
- Συχνά ευρετήριο χτίζεται αυτόματα σε αυτό

- **Foreign key:** Ένα σύνολο γνωρισμάτων σε μία σχέση, των οποίων οι τιμές **πρέπει** να ταιριάζουν με τις τιμές ενός candidate key σε μία άλλη σχέση

  Συνήθως, λόγω της μεθόδου μετατροπής, τα foreign keys αντιστοιχίζονται στο Primary Key μιας σχέσης

  Π.χ.:

  DEP (DEP#, DEPNAME, MGRE#, BUDGET)

  EMP (EMP#, ENAME, DEPT#, SALARY)

  (DEPT# και MGRE# είναι foreign keys στα DEP# και EMP#, αντίστοιχα)

Περιορισμοί (Constraints) σε (RDM)

- **Entity Integrity Constraint:**

  Δεν επιτρέπονται null τιμές σε γνωρίσματα του πρωτεύοντος κλειδιού

- **Referential integrity constraint:**

  ένα foreign key πρέπει να αντιστοιχίζεται σε κάποιο primary key/unique γνώρισμα ή να είναι εξ ολοκλήρου null.

Περιορισμοί (Constraints) σε (RDM)

- Keys στο SQL μπορούν να δηλωθούν με το **PRIMARY KEY** ή το **UNIQUE**.
- Μόνο ένα primary key, πιθανά πολλά unique.
- Primary Keys: δε μπορούν να έχουν μη ορισμένες τιμές (NULLs)
- Foreign key αναφέρονται σε Candidate Keys
- Τα κλειδιά μπορούν να οριστούν μετά τον τύπο του γνωρίσματος αν το γνώρισμα είναι απλό κλειδί

Περιορισμοί (Constraints) σε (RDM)

- Τα κλειδιά στην SQL **δηλώνονται ξεχωριστά αν περιέχουν παραπάνω από 1 γνώρισμα**.

  Πχ:

  CREATE TABLE Sells (

    bar CHAR (20),

    beer VARCHAR (20),

    price real,

    PRIMARY KEY (bar, beer))

Περιορισμοί (Constraints) σε SQL

- **NOT NULL:** κάθε πλειάδα (tuple) πρέπει να έχει ορισμένη τιμή για αυτό το γνώρισμα
  - Γενικώς, γνωρίσματα που δεν ανήκουν στο primary key μπορούν να έχουν άγνωστες/μη ορισμένες τιμές (NULLs).
  - Αυτό ισχύει και σε UNIQUE
  - Μπορούμε αν θέλουμε να ορίσουμε NOT NULL UNIQUE

- **DEFAULT:** προκαθορισμένη τιμή για το γνώρισμα, αν η τιμή δε δοθεί κατά την εισαγωγή δεδομένου

Περιορισμοί (Constraints) σε SQL

Πχ: phone CHAR (10) **NOT NULL**

… εισαγωγές δε γίνονται αν δεν οριστεί τιμή για το τηλέφωνο

Πχ: phone CHAR (10) **DEFAULT** ‘3082164803’

…εάν δε δοθεί τιμή, δώσε την τιμή 3082164803 στην εισαγωγή.

Constraints στην SQL

- **Foreign Keys στην SQL**

- Create table Beers (name char(20) Primary Key,
  manf char(20))

- Create table Sells (bar char(20),
  barA int,
  beer char(20) **references** Beers (name),
  price real,
  Primary Key (bar, beer,barA),
  FOREIGN KEY (bar, barA) references Bar(id,A))

✓ **Το foreign key (Beers(name)) πρέπει να είναι υποψήφιο κλειδί (primary key ή unique - συνήθως primary key) της άλλης σχέσης**

✓ Αν σύνθετο foreign key, το αναφέρουμε στο τέλος…:
FOREIGN KEY (FK1, FK2) REFERENCES parentB(PK1, PK2)
Foreign key (beer) references Beers(name)

Εισαγωγή στο Sells αποτυγχάνει χωρίς αντίστοιχη τιμή στο Beers

Constraints στην SQL

Επιλογές αν διαγράψουμε μία πλειάδα (tuple) στο Beers η οποία αναφέρεται από κάποια πλειάδα στο Sells ?

- **Μην το επιτρέψεις** ? → default
- Διέγραψε και αντίστοιχα tuples στο sells? → **cascade**
- Άλλαξε πεδία των tuples στο Sells σε **null**? → **set null**

In SQL2:

ON [DELETE, UPDATE] [CASCADE, SET NULL]

Επιτρέπεται διαφορετική στρατηγική για DELETE, UPDATE:

create table *course* (

. . .

foreign key (*dept name) references department*

on delete set null on update cascade,

. . .

)

Αντιστοίχιση UML Classes σε RDM

- Όμοια με τις ισχυρές οντότητες:
  - Δημιουργούμε μία σχέση (πίνακα) για την κλάση

Courses

dept	number	hours	room

Courses	
dept	PK
number	PK
hours	
room	

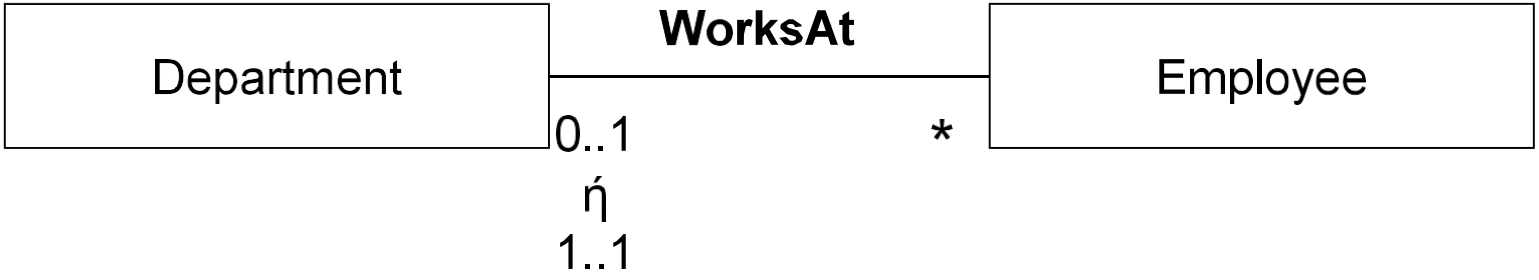
```
CREATE TABLE Courses (
        dept    VARCHAR (20),
        number int,
        hours   int,
        room    CHAR(10),
        PRIMARY KEY (dept, number)
)
```

Αντιστοίχιση UML Associations σε RDM

- Παρόμοια με τις συσχετίσεις στο E/R
- Μπορούμε να φτιάξουμε ένα πίνακα για κάθε συσχέτιση
  - Με τα κλειδιά των κλάσεων που συνδέει

- Είναι όμως απαραίτητοι όλοι αυτοί οι πίνακες;
  - Εξαρτάται από την πληθικότητα στο association…
  - Σας θυμίζει κάτι;

UML Associations σε RDM
1 προς Πολλά (παρόμοια το 1-1)




```
Create table department
    (deptno integer primary key,
     …
);

Create table employee (
  empid char (10) Primary Key,
  dept integer references department(deptno),
  … )
```

Μπορεί να πάρει null τιμές αν 1..1;
Τι γίνεται αν 0..1;

NOT NULL ????

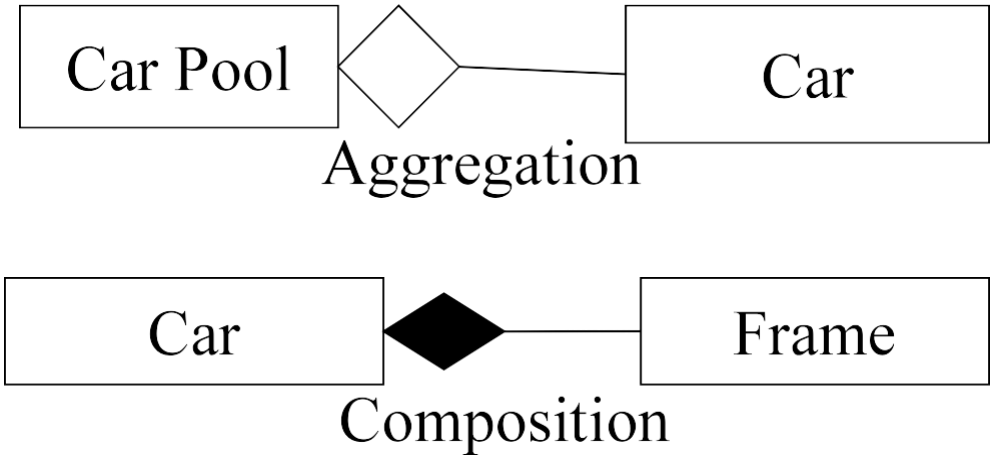
Αν 1..1, κάθε employee ΠΡΕΠΕΙ να εργάζεται σε Department, άρα το NOT NULL πρέπει να προστεθεί

Aggregation & Composition σε RDM

Θυμηθείτε:

Aggregation: 0..1 προς *

Composition: 1..1 προς *

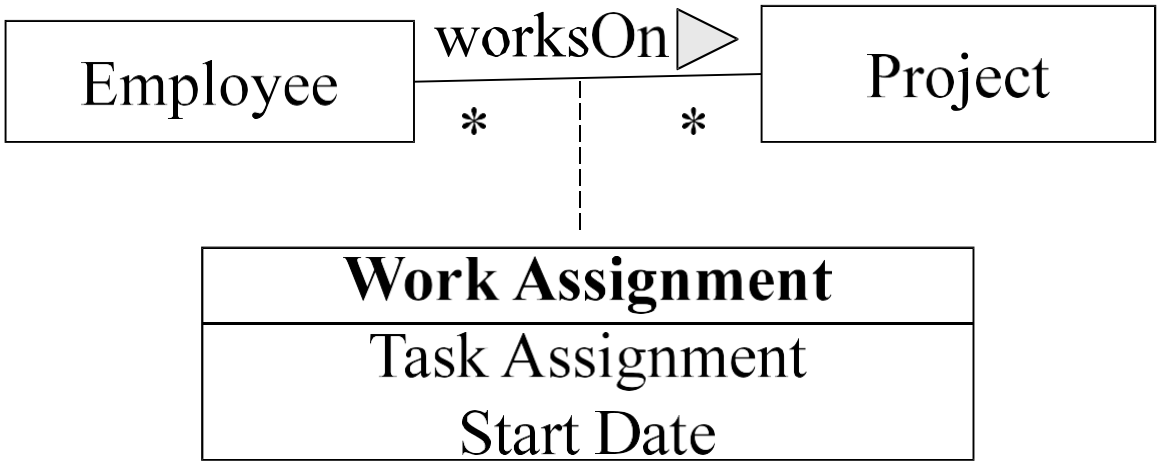



```
Create table Car
     (pinakida CHAR(7) primary key,
       pompi int references CarPool(id), …
);
Create Table Frame (
  fid int Primary Key,
  carNum CHAR(7) NOT NULL references Car(pinakida)
                on delete cascade on update cascade,
  … )
```

Ας δούμε και Association Classes

- Τι κάναμε στο E/R με τα γνωρίσματα σε συσχετίσεις;
- Αν δημιουργήσουμε νέο πίνακα για το association, προσθέτουμε σε αυτό και τα attributes του Association Class
- Αν ΔΕ δημιουργήσουμε νέο πίνακα για το association (προηγούμενη διαφάνεια), προσθέτουμε και τα attributes του Association Class στον πίνακα με το foreign key

Ας δούμε και Association Classes



Πολλά προς πολλά => νέα σχέση (πίνακας)

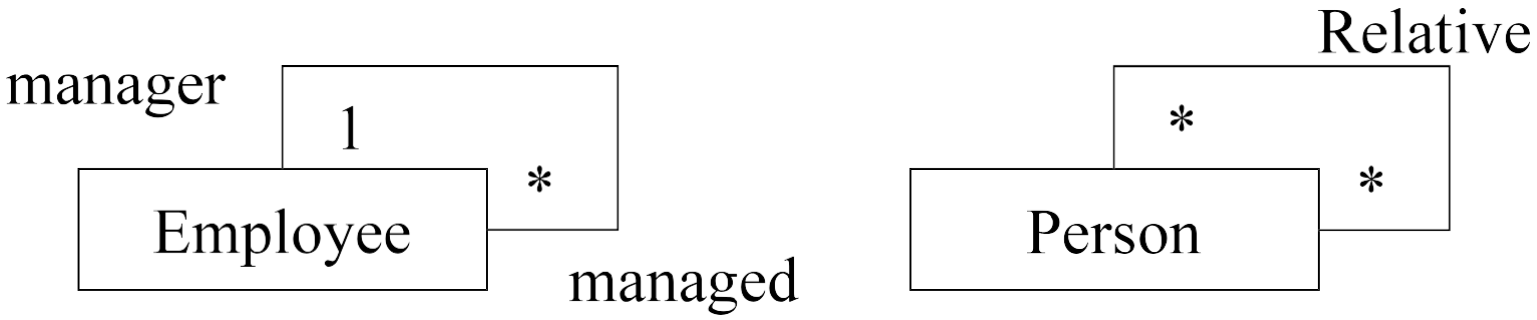
Τι θα περιέχει;

- κλειδί του Employee
- κλειδί του Project
- TaskAssignment
- StartDate

Ποιο είναι το κλειδί της νέας σχέσης;

Σύνθετο κλειδί, αποτελούμενο από τα attributes που αντιστοιχούν στο κλειδί των 2 κλάσεων Employee + Project

Αναδρομικές Associations σε RDM



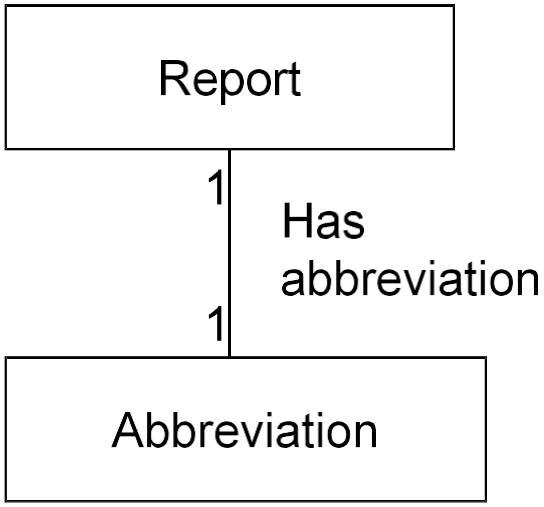
Σκεφτείτε ότι είναι κανονικά associations;
Σε ποια περίπτωση θα φτιάχνατε νέα σχέση (πίνακα);

```
Create table Relative
      (per1 integer references Person(id),
       per2 integer references Person(id),
       primary key (per1, per2))
Create table Employee (
  empid char (10) Primary Key,
  manager char(10) NOT NULL  references Employee(empid),    … )
```

Mapping UML Classes to RDM

- **One to One Associations both mandatory:**

```
Create table report
   (report_no integer,…
    primary key (report_no));

Create table abbreviation
   (abbno char (6),
    repno integer not null unique, ….
    primary key (abbno),
    foreign key (repno) references report
    on delete cascade on update cascade);
```

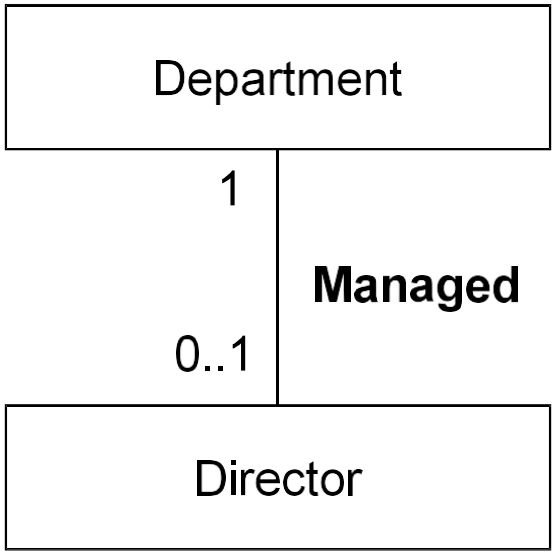


Mapping UML Classes (and ER Schema) to RDM

- Only the UML classes are presented here
- **One to One, one side optional one mandatory**

```
Create table department
     (deptno integer primary key,
      name varchar(20), …
     );

Create table Director
      (id char (10) primary key,
       dept integer references department(deptno)
                    on delete cascade on update cascade
      …,)
```

NOT NULL

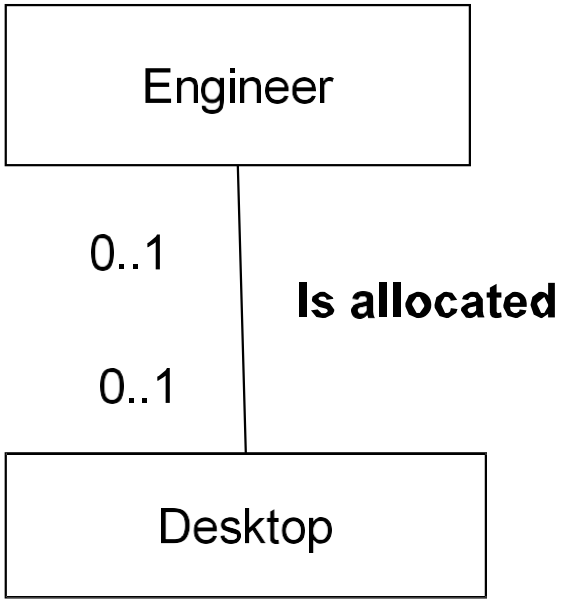


Mapping UML Classes to RDM

- **One to One both optional**

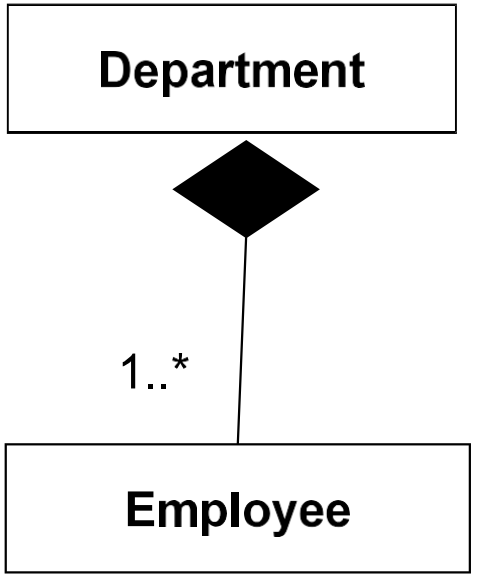
Create table engineer
(empid char (10), ….
primary key (empid));

Create table desktop
(desktopno integer,
empid char (10),
primary key (desktopno),
foreign key (empid) references engineer
on delete set null on update cascade);



Mapping UML Classes to RDM

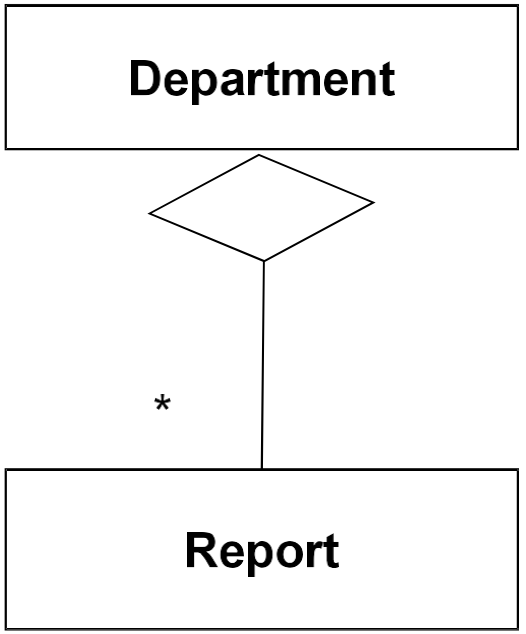
- **One to Many, both mandatory**




```
Create table department
    (deptno integer,…
     primary key (deptno));

Create table employee
     (empid char (10), …
      deptno integer not null,
      primary key (empid),
      foreign key (deptno) references department
      on delete set default on update cascade);
```

Mapping UML Classes to RDM

- **One to Many, one side optional many side mandatory**

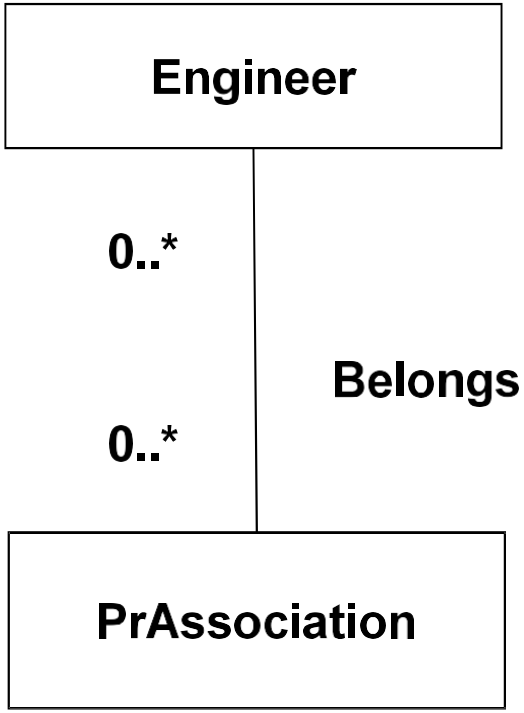



```
Create table department
    (deptno integer, …
     primary key deptno));

Create table report
    (reportno integer,
     depno integer,
     primary key (reportno),
     foreignkey (deptno) references department
     on delete set null on update cascade);
```

Mapping UML Classes to RDM

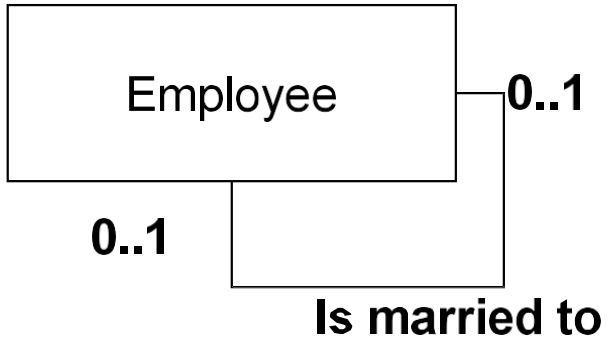
- **Many to Many both optional**

```
Create table engineer
    (empid char (10),
     primary key (empid));
Create table prassociation
    (assocname varchar (256),
     primary key (assocname));
Create table belongsto
    (empid char (10),
     assocname varchar (256),
     primary key (empid, assocname),
     foreign key (empid) references engineer,
      on delete cascade on update cascade,
     foreign key (assocname) references prassociation
      on delete cascade on update cascade
```



Mapping UML Classes to RDM

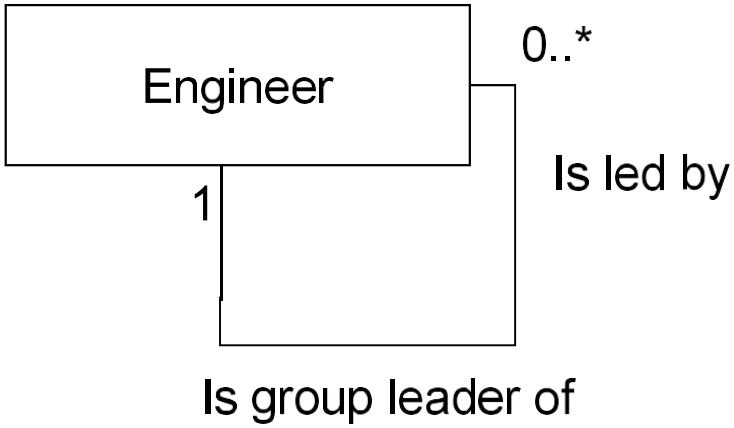
- **Recursive, one to one, both sides optional**




```
Create table employee
    (empid char (10),
     spouseid char (10), …
     primary key (empid),
     foreign key (spouceid) references employee
       on delete set null on update cascade);
```

Mapping UML Classes to RDM

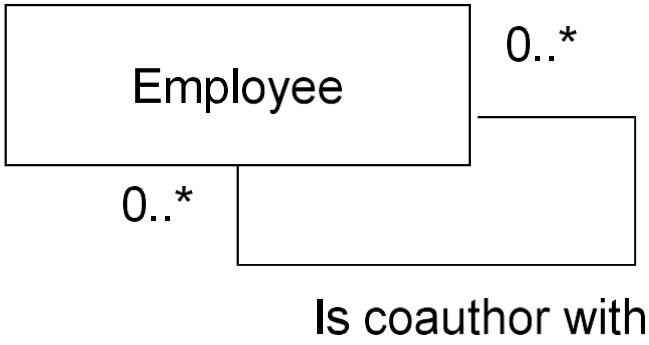
- **Recursive one to many, one side mandatory, many side optional**




```
Create table engineer
   (empid char (10),
    leaderid char (10) not null,
    primary key (empid),
    foreign key (leaderid) references engineer
      on delete set default on update cascade);
```

Mapping UML Classes to RDM

- **Recursive, many to many, both optional**



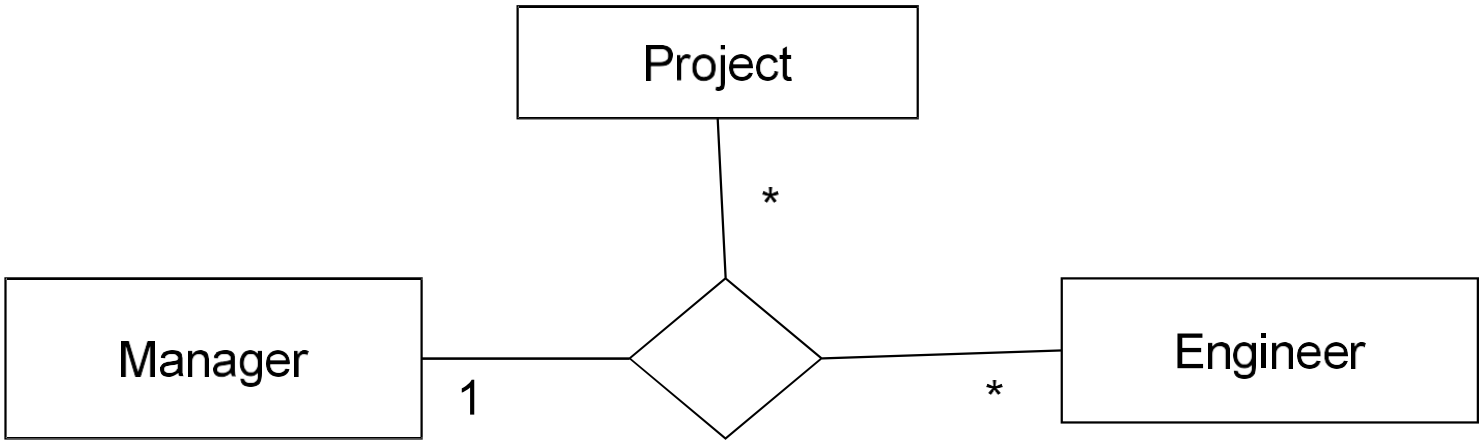

```
Create table employee
    (empid char (10),…,
     primary key (empid));

Create table coauthor
     (authorid char (10),
      coauthorid char (10),
      primary key (authorid, coauthorid),
      foreign key (authorid) references employee
          on delete cascade on update cascade,
      foreign key (coauthorid) references employee
          on delete cascade on update cascade);
```

Mapping UML Classes to RDM

- **Ternary relationships, one to many to many**

Example: One engineer has one manager in a specific project (but different managers in different projects)

Functional Dependanciy: projectname and empid determine mgrid



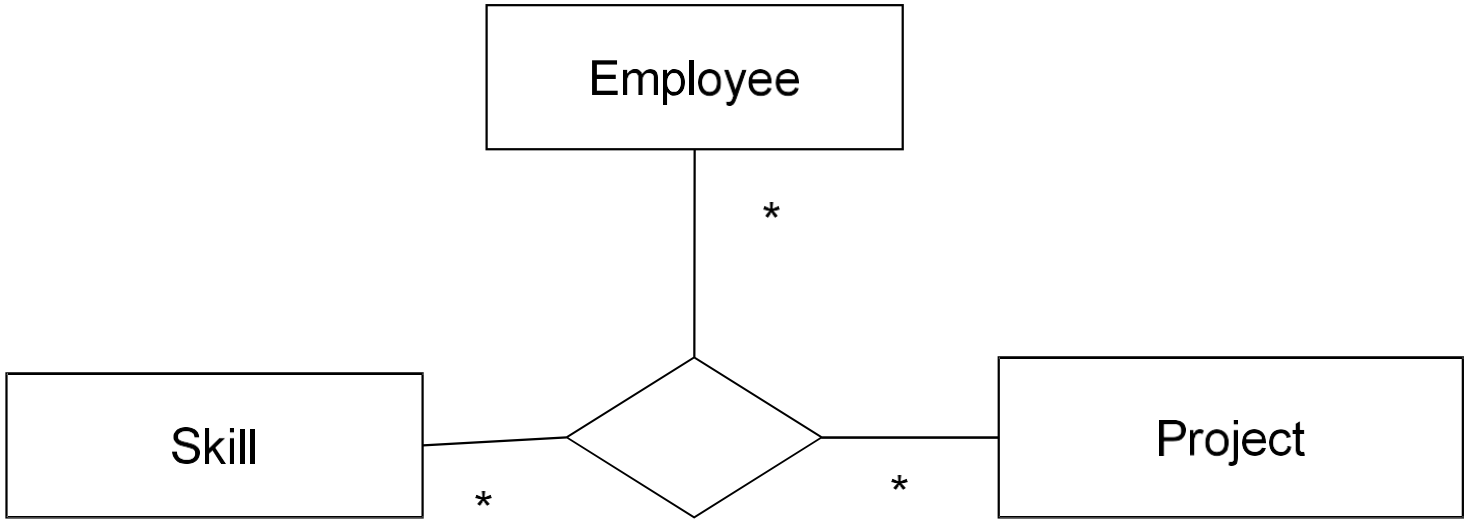
Mapping UML Classes to RDM

```
Create table project (projectname char (20),
    primary key (projectname));
Create table manager (mgrid char (10),
    primary key (mgrid));
Create table engineer (empid char (10),
    primary key (empid));
Create table manages (projectname char (20),
    mgrid char (10) not null,
    empid char (10),
    primary key (projectname, empid),
    foreign key (projectname) references project
      on delete cascade on update cascade,
    foreign key (mgrid) references manager
      on delete cascade on update cascade,
    foreign key (empid) references engineer
      on delete cascade on update cascade);
```

Mapping UML Classes to RDM

- **Ternary relationships many to many to many**

Example: employees can use many skills to ech project that they participate



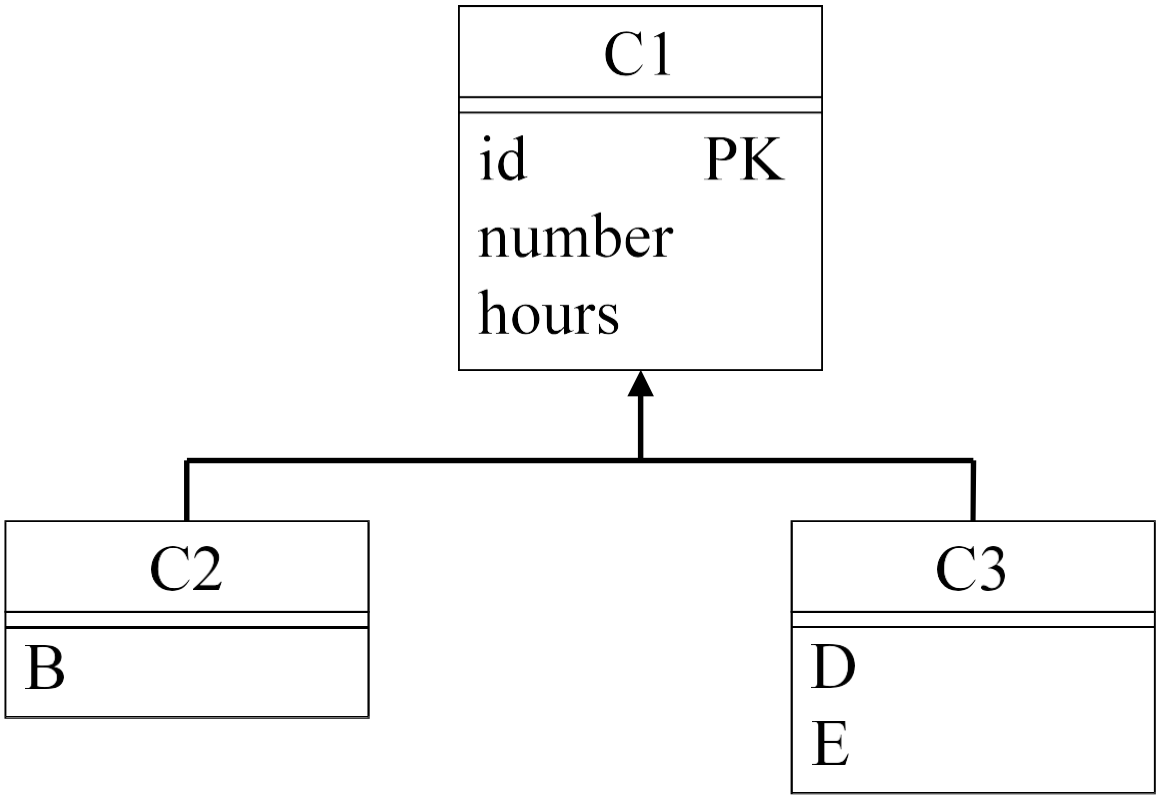
Mapping UML Classes to RDM

```
Create table employee (empid char (10),…
    primary key (empid));
Create table skill (skilltype char (15),
    primary key skilltype));
Create table project (projectname char (20),
     primary key projectname
Create table skill used (empid char (10),
      skilltype char (15),
      projectname char (20),
      primary key (empid, skilltype, projectname),
      foreign key (empid) references employee
         on delete cascade on update cascede,
      foreign key (skilltype) references skill
         on delete cascade on update cascade,
      foreign key (projectname) references project
         on delete cascade on update cascade);
```

Αντιστοίχιση Ιεραρχιών σε RDM

- **Γενίκευση (Generalization):** πολλές πιθανές στρατηγικές, καμία πάντα βέλτιστη:
  - Αντιστοίχισε **ολόκληρη την ιεραρχία σε 1 μεγάλο πίνακα**
    - **Null τιμές σε attributes που δεν έχει 1 οντότητα**
  - Δημιουργία **πίνακα για μόνο τις κλάσεις-φύλλα**
    - Χρήσιμο για disjoint και complete
  - **Κάθε κλάση σε ξεχωριστό πίνακα** (διατηρούμε το κλειδί σε όλη την ιεραρχία)
    - 2 παραλλαγές: Ένα subclass αποθηκεύει όλα τα attributes της ιεραρχίας, ή μόνο το κλειδί της υπερκλάσης;

Παράδειγμα



Επιλογή 1: C1(id, number, hours, B, D, E)
Επιλογή 2: C2(id, number, hours, B), C3(id, number, hours, D, E)
Επιλογή 3α: C1(id, number, hours), C2(id, number, hours, B),
C3(id, number, hours, D, E)
Επιλογή 3β: C1(id, number, hours), C2(id, B), C3(id, D, E)

Ανάλογα με τα χαρακτηριστικά της κληρονομικότητας, συχνά θα καταλήξετε στην επιλογή 3β ή στην επιλογή 2