

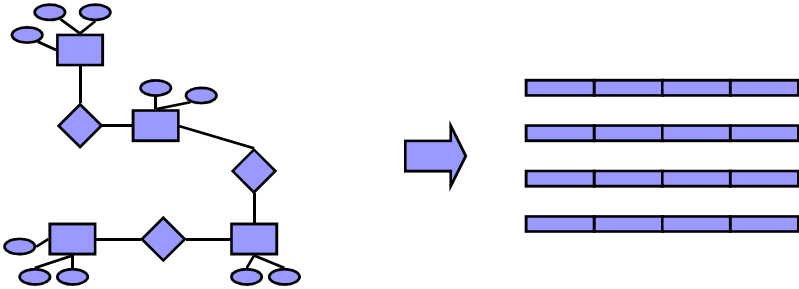
Database technology

Lecture 4: Mapping of EER model to relations

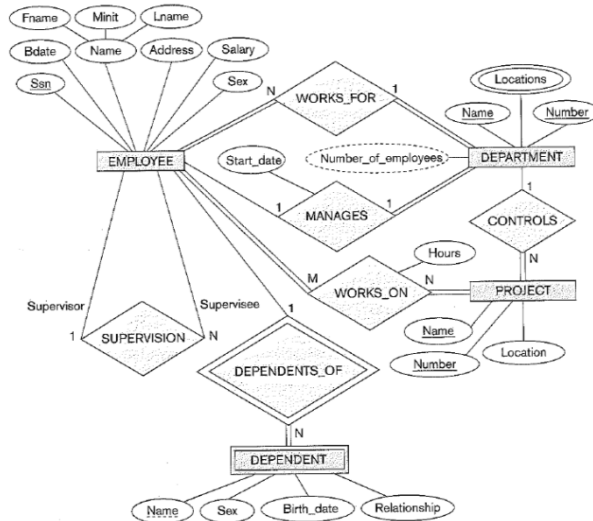
Jose M. Peña  
jose.m.pena@liu.se

## Translation ER/EER to relational

- Migrate from mini world model to a model understandable by a DBMS.



## EER model for the COMPANY database



**Figure 3.2**  
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

## ER to relations

### Step 1: Mapping regular entity types

For each strong entity type  $R$ , create a relation  $E$  that has the same simple attributes as  $R$ .



- Derived attributes are not stored.
- Composite attributes are not stored, their component ones are stored.
- Multivalued attributes are treated later.

**PROJECT**( Number, Name, Location)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, ...)

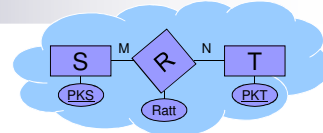
Composite attributes are not stored, their component ones are stored.

**DEPARTMENT** ( Number, Name)

"Number\_of\_employee": derived attribute are not stored.

"Location": multivalued attributes are treated later.

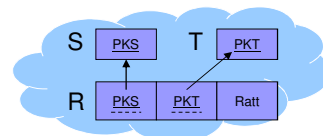
## ER to relations



### Step 5: Mapping M:N relationship types

For each binary M:N relationship, identify the relations S and T that correspond to the connected entity types. Create a new relation R and use the primary keys from S and T as foreign keys and primary keys in R. If there are attributes on the relation these are also added to R.

**On delete/update CASCADE ?!**



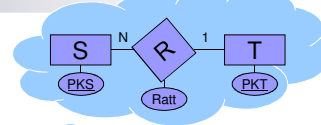
**DEPARTMENT**( Number, Name)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, ...)

**PROJECT**( Number, Name, Location)

**WorksOn**( Ssn, Number, Hours)

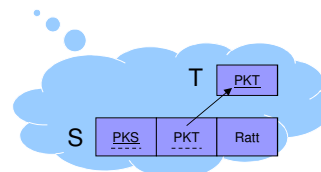
## ER to relations

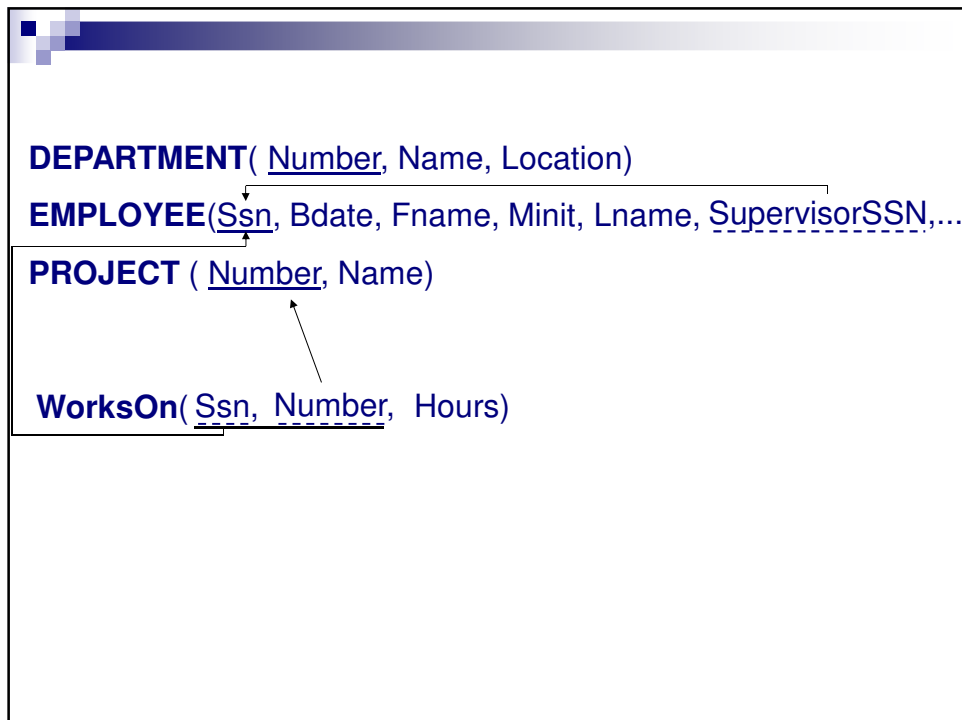


### Step 4: Mapping 1:N relationship Types

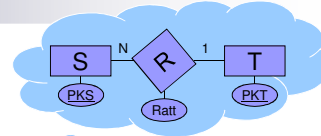
1. For each binary 1:N relationship, identify the relation S that represents the entity type on the *N-side* of the relationship type, and relation T that represents the entity type on the *1-side* of the relationship type. Include as a foreign key in S the primary key of T. If there are attributes on the relation these are also added to S.

On delete/update CASCADE ?!





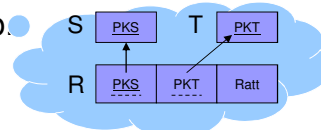
## ER to relations

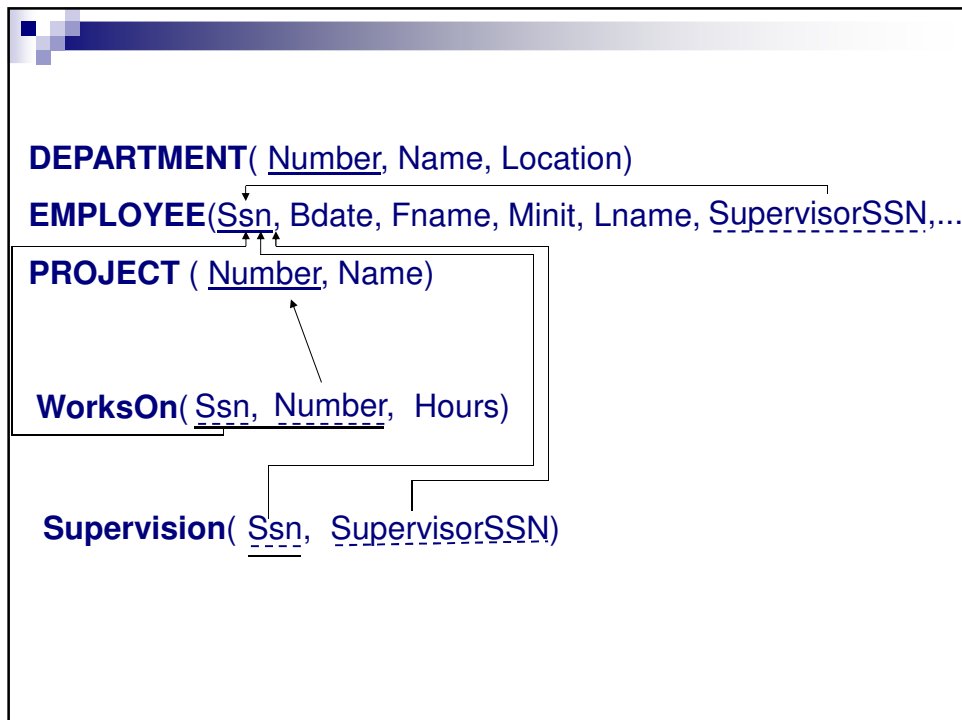


### Step 4: Mapping 1:N relationship types

1. For each binary 1:N relationship, identify the relation S that represents the entity type on the *N-side* of the relationship type, and the relation T that represents the entity type on the *1-side* of the relationship type. Include as a foreign key in S the primary key of T. If there are attributes on the relation these are also added to S.
2. Implement as a M:N relationship (unlike M:N relationship, now **PK(R) is PK(S)**). Convenient if few tuples participate in the relationship.

On delete/update **CASCADE** ?!





## ER to relations

### Step 3: Mapping 1:1 relationship types

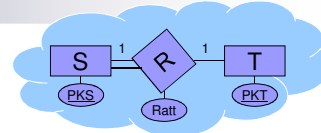
1. Implement as a 1:N relationship (prefer the entity type with **total participation**, if any, as the entity type to which the foreign key is added).

**PROJECT**( Number, Name, Location)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, ...)

**DEPARTMENT** ( Number, Name, Manager )

## ER to relations



### Step 3: Mapping 1:1 relationship types

1. Implement as a 1:N relationship (prefer the entity type with **total participation**, if any, as the entity type to which the foreign key is added).
2. For each binary 1:1 relationship B, identify the relations S and T that correspond to the incoming entity types. Merge S and T into a single relation R. Set the primary key of S or T as the primary key of R. Do not forget the attributes of the relationship type. **Indicated only when S and/or T with total participation.**



## ER to relations

### Step 2: Mapping weak entity types

For each weak entity type **W** with owner entity type **E**, create a relation **R** that has the same simple attributes as **W**, also add (as a foreign key) the primary key attributes from the relation that corresponds to **E**. The primary key attributes in R are composed of the primary key attributes from E and the partial key from W.

On delete/update CASCADE ?!

**DEPARTMENT**( Number, Name)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, SupervisorSSN, ...)

**PROJECT** ( Number, Name)

**WorksOn**( Ssn, Number, Hours)

**DEPENDENT**( Ssn, Name, Sex, Birth\_date, ...)



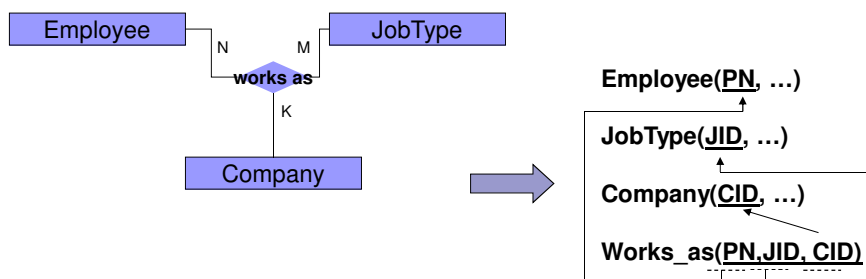
## ER to relations

### Step 7: Mapping N-ary relationship types

For each N-ary relationship with  $N > 2$ , create a new relation  $S$  that contains the primary keys from the incoming relations as foreign keys. The primary key of  $S$  are those keys that come from cardinality constraints  $\neq 1$ . Do not forget the attributes of the relationship type.

On delete/update CASCADE ?!

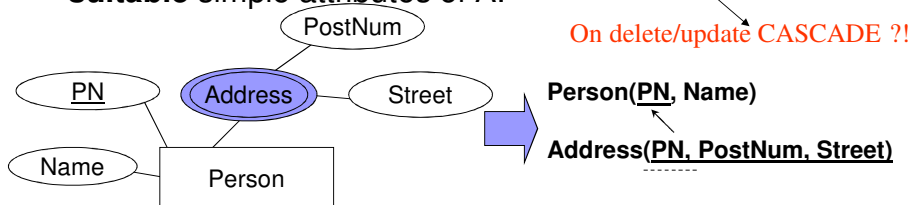
- Example. A person works as an engineer at one company and as a gym instructor at another company.



## ER to relations

### Step 6: Mapping multivalued attributes

For each multivalued attribute  $A$  in  $R$ , create a new relation  $P$  that contains one attribute for each attribute in  $A$  and the primary key  $K$  of  $R$  as a foreign key. The primary key of  $P$  is the combination of  $K$  and **some suitable** simple attributes of  $A$ .



## ER to relations

### ■ Materializing the relationship:

- M:N implies two joins.
- 1:N implies one or two joins.
- 1:1 implies zero, one or two joins.
- N-ary implies N joins.

**DEPARTMENT**( Number, Name)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, ...)

**PROJECT**( Number, Name, Location)

**WorksOn**( Ssn, Number, Hours)

```

SELECT E.Fname, P.Name, W.Hours
FROM EMPLOYEE E, PROJECT P, WorksOn W
WHERE W.SSN = E.SSN AND W.Number = P.Number
  
```

**PROJECT**( Number, Name, Location)

**EMPLOYEE**(Ssn, Bdate, Fname, Minit, Lname, ...)

**DEPARTMENT** ( Number, Name, Manager)

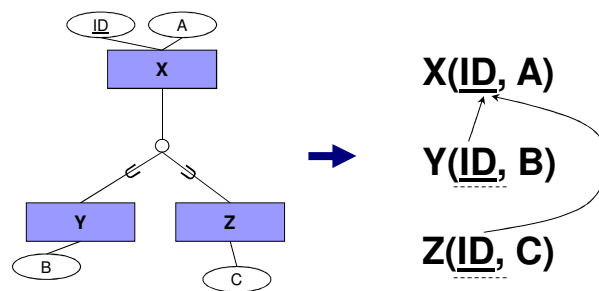
```

SELECT E.Fname, D.Name
FROM EMPLOYEE E, DEPARTMENT D
WHERE D.Manager = E.Ssn;
  
```

## EER to relations

### Step 8: Mapping specialization

- a) Create relations for each class (super+subclass).

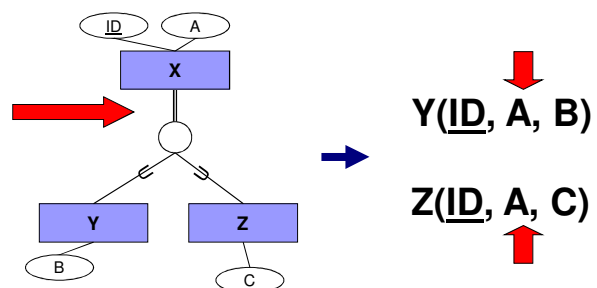


\* Always works.

## EER to relations

### Step 8: Mapping specialization

- b) Create relations for the subclasses only.

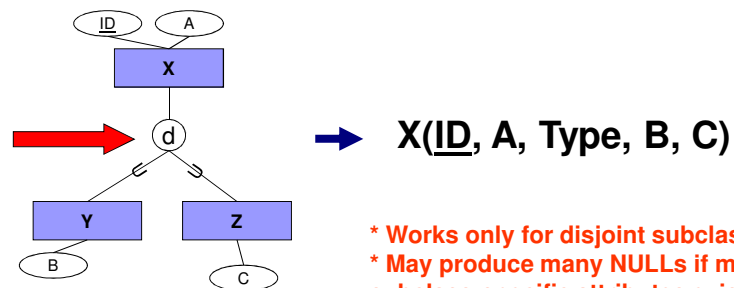


\* Works only for total participation.  
\* Overlapping implies duplication.

## EER to relations

### Step 8: Mapping specialization

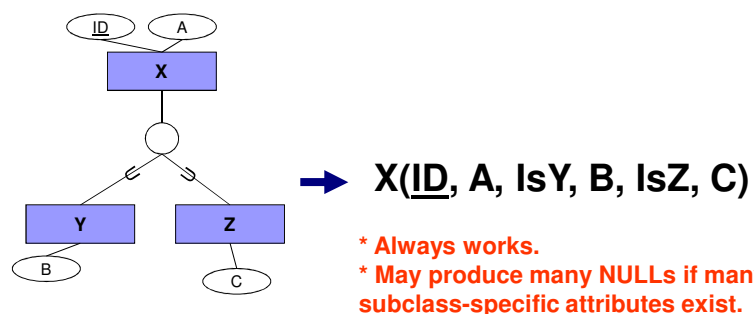
- c) Create a single relation with one type attribute and all subclass attributes.



## EER to relations

### Step 8: Mapping specialization

- d) Create a single relation with multiple type attributes and all subclass attributes.



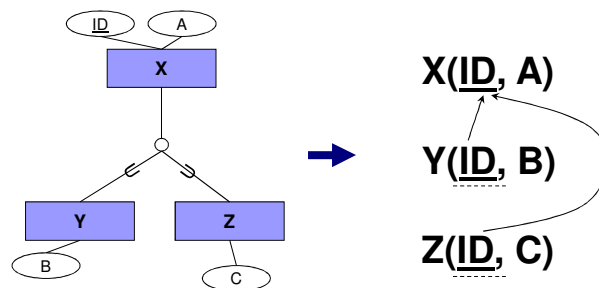
## EER to relations

### ■ Materializing the superclass/subclasses:

- Option a, inner/outer join.
  - Option b, outer join (against theory...).
  - Option c, done.
  - Option d, done.
- } May be more space inefficient but more time efficient.

## EER to relations

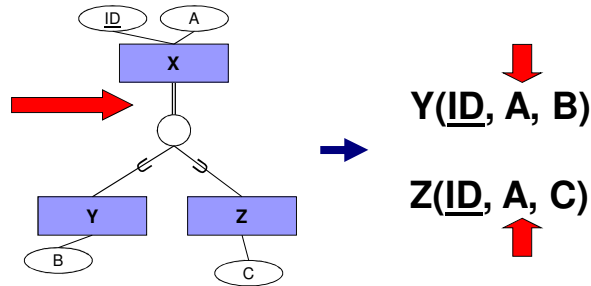
a) Create relations for each class (super+subclass).



```
SELECT X.ID, X.A, Y.B
FROM X LEFT JOIN Y ON X.ID = Y.ID;
```

## EER to relations

- b) Create relations for the subclasses only.

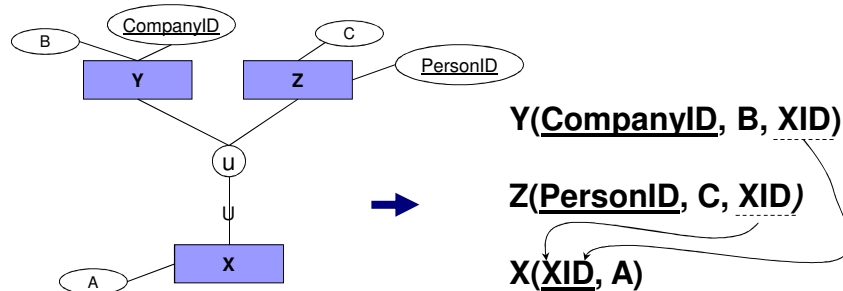


```
SELECT Y.ID, Z.ID, Y.A, Z.A, Y.B, Z.C
FROM Y FULL OUTER JOIN Z ON Y.ID = Z.ID;
```

## EER to relations

### Step 9: Mapping of union types

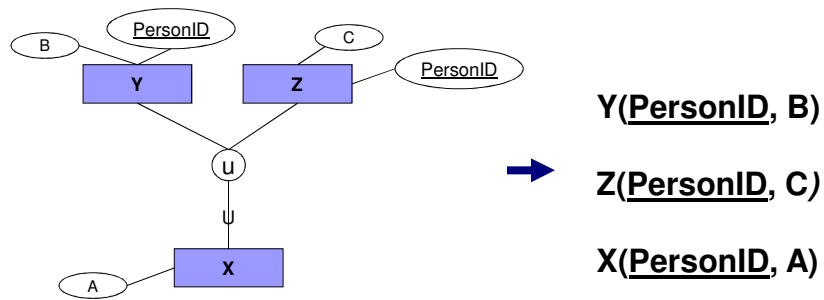
- a) If the defining superclasses have different primary keys, introduce a surrogate key in the union relation and use it as a foreign key in the superclasses.



# EER to relations

## Step 9: Mapping of union types

- b) If the defining superclasses use the same primary key, no need for surrogate key.



\* No FKs in Y and Z, unless total participation (correct figure 7.7 in the book)

## Example: LARM days

