



## **COSC210 Database Management Systems**

# **Lecture 7 - Relational Database Design by ER/EER-to-Relational Mapping**

**Dr. Mitchell Welch / Dr. Edmund Sadgrove**

## Reading

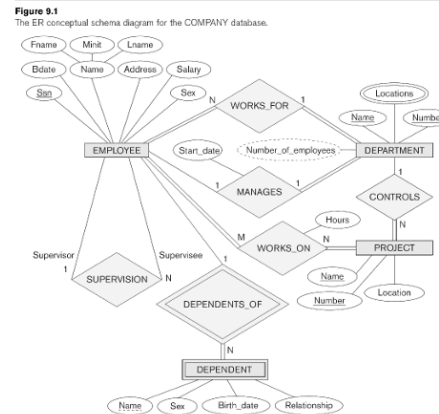
- Chapter 9 from *Fundamentals of Database Systems* by Elmazri and Navathe

## Summary

- ER-to-Relational Mapping Algorithms
- SQL Implementation
- EER-to-Relational Mapping Algorithms

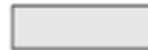
## ER-to-Relational Mapping Algorithms

- Mapping - Step One:
  - **Strong entities:**
    - Entity type  $E$  is used to create a relation  $R$ .
    - Includes only:
      - Simple/single-valued attributes of  $E$ .
  - **Primary key:**
    - Selected from key attributes of  $E$ .
    - Can be a **composite key**.
  - **Candidate keys:**
    - Other key attributes listed as candidates.
    - Useful for indexing.



## ER-to-Relational Mapping Algorithms

- Mapping - Step Two:
  - **Weak entity:**
    - Weak entity type  $W$  is mapped to a relation  $R$ .
    - Includes only:
      - Simple/single-valued attributes of  $E$ .
  - **Primary key:**
    - Combination of owner and **partial key**.
  - **Foreign key:**
    - Owner primary key.
    - Constraints:
      - ON DELETE CASCADE.
      - ON UPDATE CASCADE.



Entity

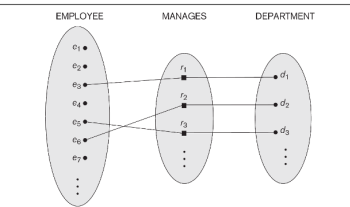


Weak Entity

## ER-to-Relational Mapping Algorithms

- Mapping - Step Three:
  - **1:1** relationships  $R$ :
    - Identify the participating relations  $S$  and  $T$ .
  - Mapping has three options:
    - Using **foreign keys**.
    - **Merging** Relations.
    - An **additional** relation.

**Figure 7.12**  
A 1:1 relationship,  
MANAGES.



<sup>1</sup>N stands for any number of related entities (zero or more).

# ER-to-Relational Mapping Algorithms

- Mapping - Step Three (option 1):

- **Foreign key:**

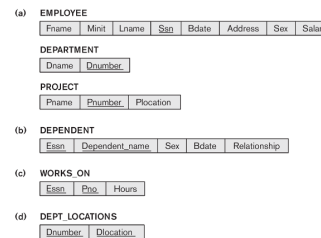
- Select one relation (say  $S$ ).
    - Include a foreign key that points to primary key of  $T$ .

- Select entity with **total participation** for role of  $S$

- Every tuple in  $S$  references a tuple in  $T$ .
    - E.g manager - employee ( $S$ ) references department ( $T$ ).
    - The reverse will cause null values (non managers in  $S$ ).

**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.



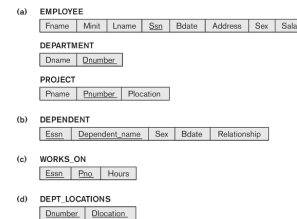
## ER-to-Relational Mapping Algorithms

- Mapping - Step Three (option 2):
  - **Merging relations:**
    - If both  $S$  and  $T$  have total participation in  $R$ .
    - Merge attributes of  $S$  and  $T$  into single relation.
- Mapping - Step Three (option 3):
  - **Additional relation:**
    - Contains tuples that map the keys from  $S$  to  $T$ .
    - Option for 1:1, but required for N:M.

## ER-to-Relational Mapping Algorithms

- Mapping - Step Four:
  - **1:N** relationship  $R$ :
    - In binary a relationship  $R$ , relation  $S$  is on the *N-side*.
  - **Foreign key:**
    - The foreign key is placed on the *N-side*.
    - Tuples in  $S$  references primary key of  $T$ .
    - As multiple tuples of  $S$  point to a single tuple of  $T$ .
    - The reverse will cause redundancy.

**Figure 9.3**  
Illustration of some  
mapping steps.  
a. Empty 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.



- An alternative approach is to use a **third relation** to map the keys from  $S$  to  $T$ .

# ER-to-Relational Mapping Algorithms

- Mapping - Step Five:

- **M:N** relationship *R*:

- Create additional weak relation *W*:

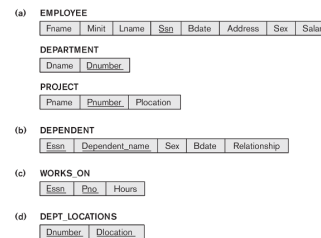
- This relation has an existence dependency.
      - **Primary key** will be a composite key of *S* and *T* primary keys.
      - Requires **foreign keys** that reference keys of *S* and *T*.

- **Constraints:**

- ON DELETE CASCADE.
    - ON UPDATE CASCADE.

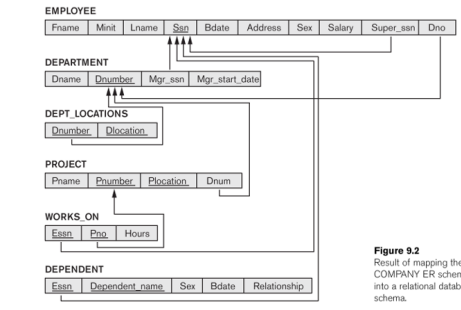
**Figure 9.3**

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.



## ER-to-Relational Mapping Algorithms

- Mapping - Step Six:
  - **Multi-valued** attribute *A* of relation *T*:
    - Create additional relation *S*:
      - Contains values from *A* as primary key tuples (single attribute).
    - **Foreign key**:
    - *T* references primary key of new relation *S*.

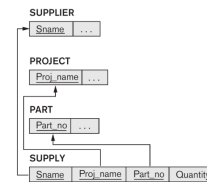


- For example: **DEPT\_LOCATIONS**
- [http://turing.une.edu.au/~csc210/workshops/prac\\_1/data/dept\\_locations.csv](http://turing.une.edu.au/~csc210/workshops/prac_1/data/dept_locations.csv)

## ER-to-Relational Mapping Algorithms

- Mapping - Step Seven (final step):
  - **N-ary** (non-binary) relation  $R$ .
    - Create a new relation  $W$ .
    - **Primary key** will be a composite key of participating relation primary keys.
    - **Foreign keys** will reference the primary keys of the participating relations.
- The new relation  $W$  will be a weak relation.

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



## SQL Implementation

- Mapping in development may still require an implementation model.
  - Such as the unified modelling language (**UML** diagram).
- We will map directly to *PostgreSQL*.
  - We need to think carefully about the operations:
    - **Referential constraints** and Triggers.
- This can be demonstrated through the implementation of the **COMPANY database**

## SQL Implementation

- The EMPLOYEE entity is represented as an SQL table.
- Notice that the composite attribute name has been decomposed into three separate columns

```
CREATE TABLE employee (  
  fname  varchar(15) NOT NULL,  
  minit  varchar(1),  
  lname  varchar(15) NOT NULL,  
  ssn     char(9) PRIMARY KEY,  
  bdate  date,  
  address varchar(50),  
  sex     char,  
  salary  decimal(10,2),  
  superssn char(9),  
  dno     integer,  
  foreign key (superssn) references employee(ssn)  
    ON DELETE SET NULL ON UPDATE CASCADE,  
  foreign key (dno) references department(dnumber)  
    ON DELETE SET NULL ON UPDATE CASCADE  
);
```

## SQL Implementation

- The DEPARTMENT and PROJECT entities can be implemented in a similar fashion.

```
CREATE TABLE department (  
  dname      varchar(25) not null,  
  dnumber    integer primary key,  
  mgrssn     char(9) not null,  
  mgrstartdate date  
);  
  
CREATE TABLE project (  
  pname      varchar(25) unique not null,  
  pnumber    integer primary key,  
  plocation  varchar(15),  
  dnum       integer not null,  
  foreign key (dnum) references department(dnumber)  
  ON DELETE SET NULL ON UPDATE CASCADE  
);
```

## SQL Implementation

- The weak entity DEPENDENT is implemented as a table with a foreign key referencing the employee table.
- Notice that the primary key consists of the essn and the dependent\_name - this represents the esi

```
CREATE TABLE dependent (  
    essn          char(9),  
    dependent_name varchar(15),  
    sex           char,  
    bdate         date,  
    relationship   varchar(8),  
    primary key (essn, dependent_name),  
    foreign key (essn) references employee(ssn)  
    ON DELETE CASCADE ON UPDATE CASCADE  
);
```

## SQL Implementation

- The WORKS\_ON table implements the M:N relationship between EMPLOYEE and PROJECT.

```
CREATE TABLE dept_locations (  
  dnumber integer,  
  dlocation varchar(15),  
  primary key (dnumber,dlocation),  
  foreign key (dnumber) references department(dnumber)  
  ON DELETE CASCADE ON UPDATE CASCADE  
);
```

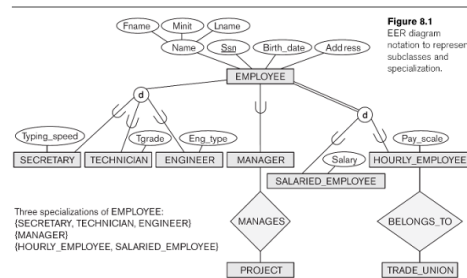
## SQL Implementation

- The DEPT\_LOCATION table implements the multi-valued department location attribute. Note the direction of the relationship.

```
CREATE TABLE works_on (  
    essn    char(9),  
    pno     integer,  
    hours   decimal(4,1),  
    primary key (essn,pno),  
    foreign key (essn) references employee(ssn)  
        ON DELETE CASCADE ON UPDATE CASCADE,  
    foreign key (pno) references project(pnumber)  
        ON DELETE CASCADE ON UPDATE CASCADE  
);
```

## EER-to-Relational Mapping Algorithms

- Recall that the EER model is an extension of the ER model.
  - This means we can map relations in the same way.
- Two mapping algorithms are added:
  - For **Specialisations/Generalisation**.
  - For **union types**.
- These are applied as steps 8 and 9.



<sup>2</sup>A class/subclass relationship is often called an **IS-A** (or **IS-AN**) relationship because of the way we refer to the concept. We say a SECRETARY is an EMPLOYEE, a TECHNICIAN is an EMPLOYEE, and so on.

## EER-to-Relational Mapping Algorithms

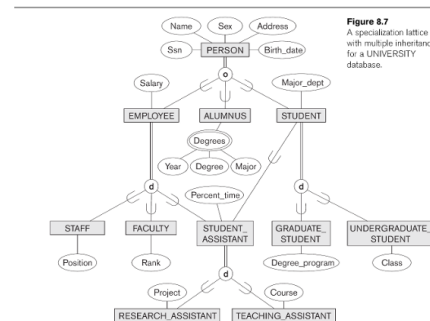
- Mapping - Step Eight:
  - Mapping entities in **Specialisation/Generalisation**.
- There are a 4 options for representing this relationship:
  - Multiple relations - **superclass and subclasses**.
  - Multiple relations - **subclasses only**.
  - Single relation using a ***type* attribute**.
  - Single relation with multiple ***type* attributes**.

## EER-to-Relational Mapping Algorithms

- Mapping - Step Eight (option 1):
  - Create relations for all super and sub classes.
  - The **superclass** relation  $L$  will have a **primary key**  $PK(L)$ .
  - The **subclass** relations  $L_1, L_2, \dots, L_n$  will have a primary key  $PK(L_i)$ .
    - Such that  $PK(L_i) = PK(L)$
    - This will be a **foreign key** that references  $L$ .
- This will adhere to superclass/subclass **constraints**.
- This approach is suitable for any specialisation (d,o, total and/or partial)

## EER-to-Relational Mapping Algorithms

- Mapping - Step Eight (option 2):
  - Create a relation for each subclass (only).
    - **Superclass** attributes are replicated within **subclass** relations.
    - Only works if the relationship is **disjoint**.
    - Not recommended for **overlapping**.
      - This will lead to further redundancy.
- This approach **involves redundancy**.

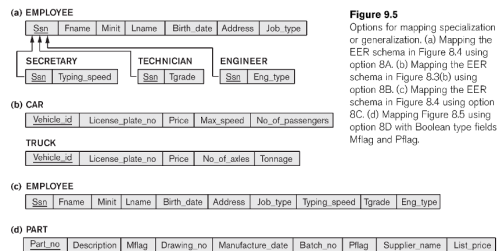


## EER-to-Relational Mapping Algorithms

- Mapping - Step Eight (option 3):
  - Create a single relation with a **type attribute**.
    - This will indicate the subclass entity that the tuple represents.
    - This will only work for **disjoint** specialisations.
- This approach will generate **NULL** values within tuples with types where **specific attributes** are not applicable.

## EER-to-Relational Mapping Algorithm

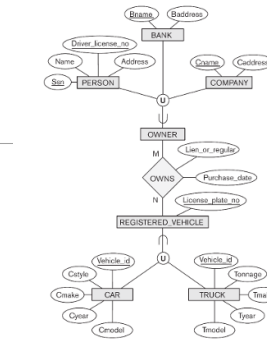
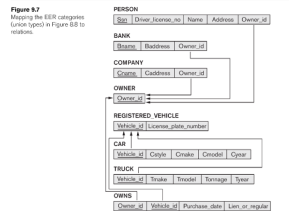
- Mapping - Step Eight (option 4):
  - Create a single relation, **boolean-type** attribute.
    - Indicates membership with true/false flag.
- This will work for **overlapping** specialisations.



**Figure 9.5**  
Options for mapping specialization or generalization. (a) Mapping the EER schema in Figure 8.4 using option 8A. (b) Mapping the EER schema in Figure 8.3(b) using option 8B. (c) Mapping the EER schema in Figure 8.4 using option 8C. (d) Mapping Figure 8.5 using option 8D with Boolean type fields Mflag and Pflag.

## EER-to-Relational Mapping Algorithms

- Mapping - Step Nine (final):
  - Union types** (recall):
    - A **Subclass** inherits all attributes and relationships from *multiple* superclasses.
  - Surrogate key**:
    - A new relation is created with a new key.
    - Surrogate key is added to superclass relations.



## Summary

- ER-to-Relational Mapping Algorithms
- SQL Implementation
- EER-to-Relational Mapping Algorithms

# Questions?

## Next Lecture

- SQL Two (Query Structure).

## Reading

- Chapters 4 and 9 from *Fundamentals of Database Systems*
- Chapter 7 from *Fundamentals of Database Systems* this week's prac session.

