



**KOÇ
UNIVERSITY**

Database Management Systems

Converting from ER to Relational Model

M. Emre Gürsoy

Assistant Professor
Department of Computer Engineering

www.memregursoy.com



Introduction

- We had modelled our **miniworld** using an **ER diagram**
- Now, we will learn how to convert **ER diagrams** to a **set of relations** (Relational Model)
- After we obtain the **set of relations**, in the next weeks we will learn how to work with relations
 - Theoretical: Relational algebra
 - Practical: Relational DBMS and SQL
- **Main goals in ER -> Relational conversion:**
 - (1) Preserve as much information as possible
 - (2) Minimize redundancy and NULL values



Conversion Algorithm

- We will follow a step-by-step algorithm.
- Step 1: Mapping of Regular Entities
 - Mapping of Subclass/Superclass
- Step 2: Mapping of Weak Entities
- Step 3: Mapping of Binary 1:1 Relationships
- Step 4: Mapping of Binary 1:N Relationships
- Step 5: Mapping of Binary M:N Relationships
- Step 6: Mapping of Multivalued Attributes
- Step 7: Mapping of N-ary Relationships (eg: ternary)



Step 1

- **Regular (non-weak) entities**
- For each regular entity **E**, create a relation **R** that includes all the simple attributes of **E**.
 - If **E** has a composite attribute, take the simple attributes that make up the composite attribute.
- Choose one of **E**'s keys as the primary key of **R**
 - If **E**'s key is composite, the set of simple attributes that make up the composite attribute COMBINED becomes the key of **R**

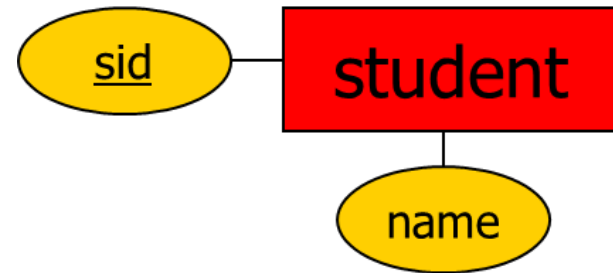
EMPLOYEE (Ssn, Bdate, Fname, Minit, Lname, Address, Salary, Sex)



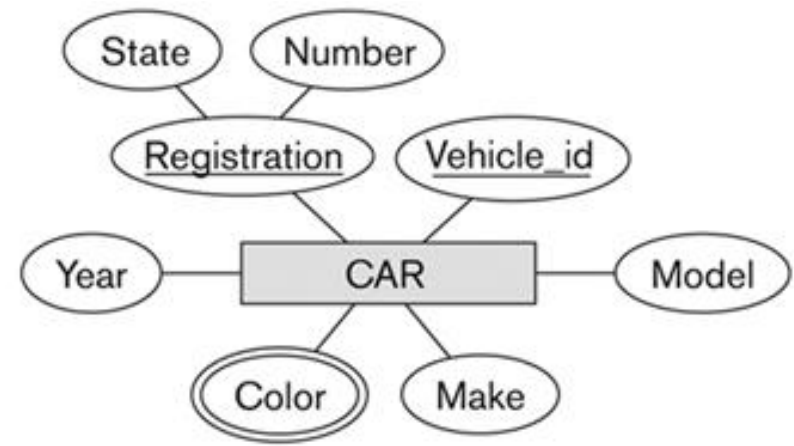


Step 1 Examples

```
CREATE TABLE Student  
(sid: CHAR(8),  
name: CHAR(30),  
PRIMARY KEY (sid))
```



```
CREATE TABLE Car  
(reg_state: CHAR(2),  
reg_number: CHAR(10),  
vehicle_id: CHAR(15),  
model: CHAR(10),  
make: CHAR(15),  
year: INTEGER,  
PRIMARY KEY (reg_state,  
reg_number))
```

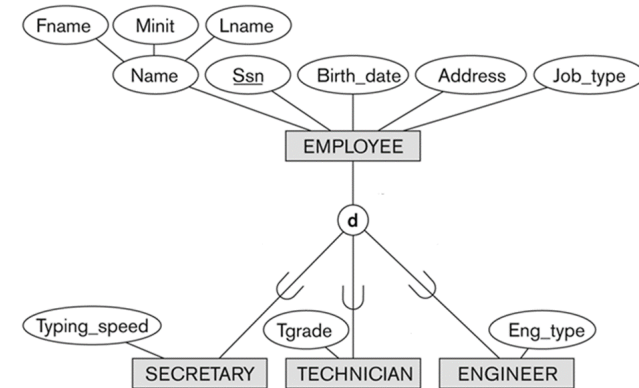




Subclasses/Superclasses

- **Subclasses/Superclasses**

- There are multiple options in the book.
 - Some better for disjoint, some better for overlapping
 - Some better with participation constraint, some not
 - We will study just one (most general) option

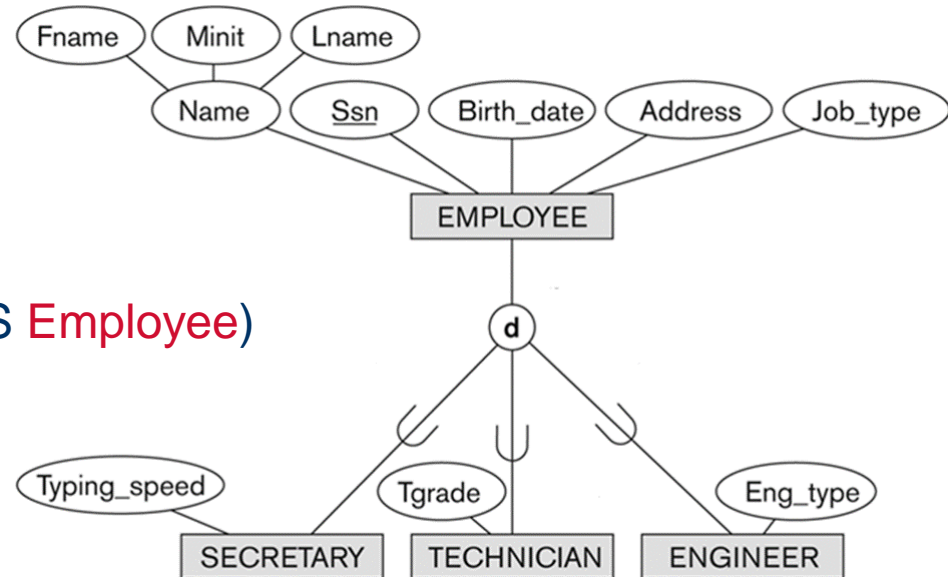


- Keep the **superclass** relation as is.
- For each **subclass**, create its own relation.
 - Add primary key of superclass to the subclass relation, as a foreign key.
 - Add any other attribute the subclass may have to the subclass relation.
 - Primary key of subclass relation is equal to the primary key of the superclass.



Examples

CREATE TABLE Secretary
(SSN: INTEGER,
TypingSpeed: REAL,
PRIMARY KEY (SSN),
FOREIGN KEY SSN
REFERENCES Employee)



EMPLOYEE

<u>SSN</u>	FName	MInit	LName	BirthDate	Address	JobType
------------	-------	-------	-------	-----------	---------	---------

SECRETARY

<u>SSN</u>	TypingSpeed
------------	-------------

TECHNICIAN

<u>SSN</u>	TGrade
------------	--------

ENGINEER

<u>SSN</u>	EngType
------------	---------



Step 2

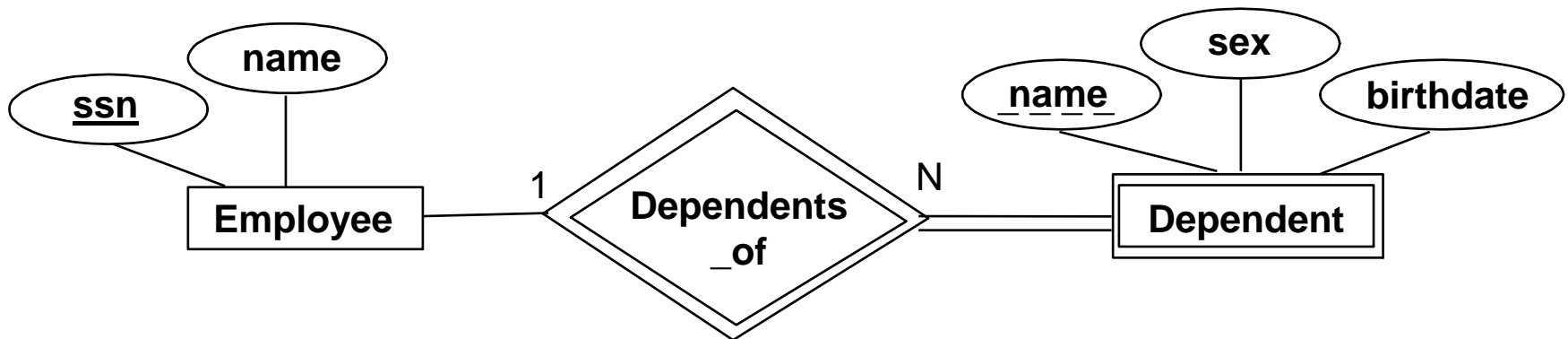
- **Weak entities (+ identifying relationships)**
- We handle the weak entity and its identifying relationship together, by creating one new relation **R**.
- The new relation **R** contains:
 - Owner (strong) entity's primary key -> foreign key in **R**
 - All attributes of the weak entity
 - All attributes of the identifying relationship (if any)
- Primary key of **R** is the **combination** of the primary key of the owner plus the partial key of the weak entity



Step 2 Examples

```
CREATE TABLE Employee  
(ssn: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (ssn))
```

```
CREATE TABLE Emp_Dependents  
(emp_ssn: INTEGER,  
dep_name: CHAR(30),  
dep_sex: CHAR(10),  
dep_birthdate: CHAR(10),  
PRIMARY KEY (emp_ssn, dep_name),  
FOREIGN KEY emp_ssn  
REFERENCES Employee)
```





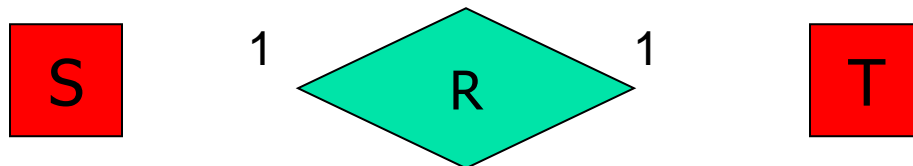
Conversion Algorithm

- We will follow a step-by-step algorithm.
- Step 1: Mapping of Regular Entities
 - Mapping of Subclass/Superclass
- Step 2: Mapping of Weak Entities
- Step 3: Mapping of Binary 1:1 Relationships
- Step 4: Mapping of Binary 1:N Relationships
- Step 5: Mapping of Binary M:N Relationships
- Step 6: Mapping of Multivalued Attributes
- Step 7: Mapping of N-ary Relationships (eg: ternary)



Step 3

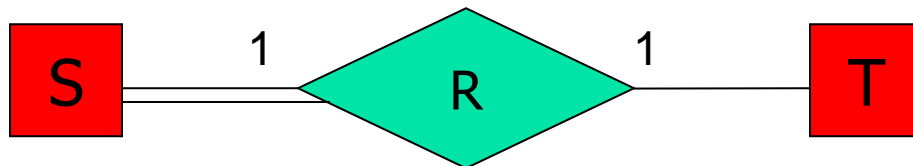
- **Binary 1-to-1 Relationships**
- There are 3 cases:
 - (3A) Participation constraint only on one side
 - (3B) Participation constraints on both sides
 - (3C) No participation constraints





Step 3-A

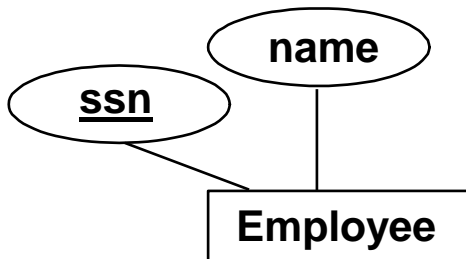
- **Binary 1-to-1 Relationships**
- **Participation constraint only on one side**
- (Note: From Step 1, we already have a relation created for **S** and one relation created for **T**.)
- Let **S** be the fully participating entity
- Add **T**'s primary key to **S**'s relation as foreign key
- Any attribute of relationship **R** is also added to **S**



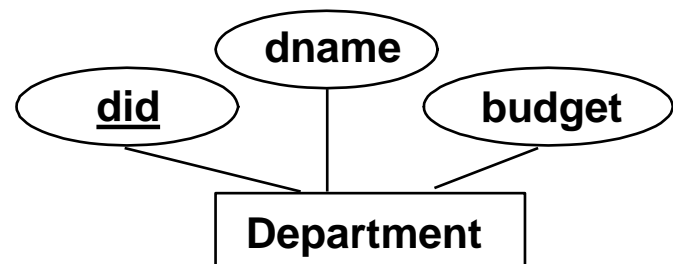


Step 3-A Examples

```
CREATE TABLE Employee  
(ssn: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (ssn))
```



```
CREATE TABLE Department  
(did: INTEGER,  
dname: CHAR(30),  
budget: REAL,  
PRIMARY KEY (did))
```

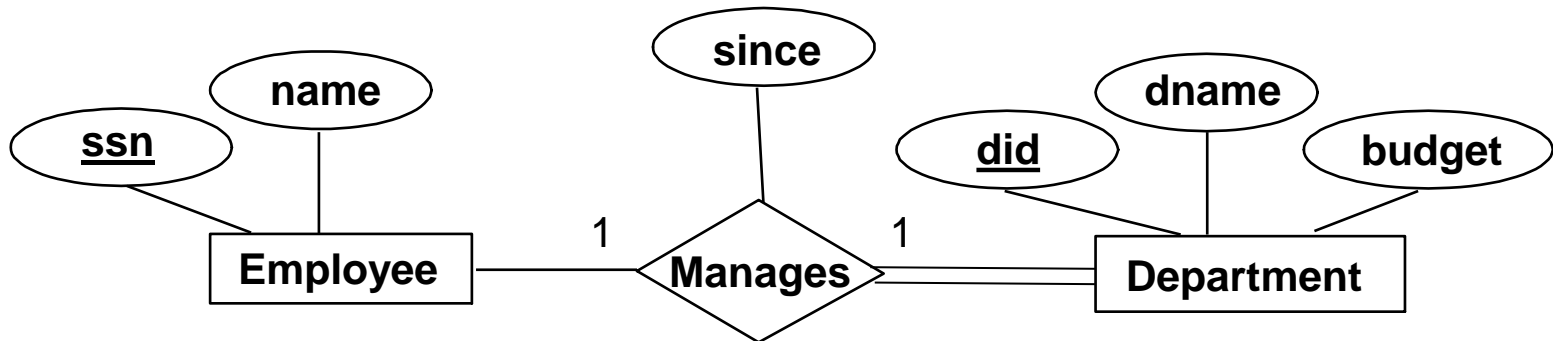




Step 3-A Examples

```
CREATE TABLE Employee  
(ssn: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (ssn))
```

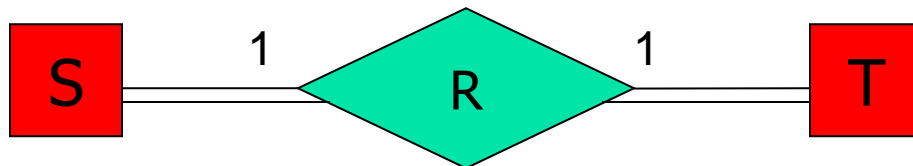
```
CREATE TABLE Department  
(did: INTEGER,  
dname: CHAR(30),  
budget: REAL,  
PRIMARY KEY (did),  
manager_ssn: INTEGER,  
FOREIGN KEY(manager_ssn)  
REFERENCES Employee  
manager_since: CHAR(10))
```





Step 3-B

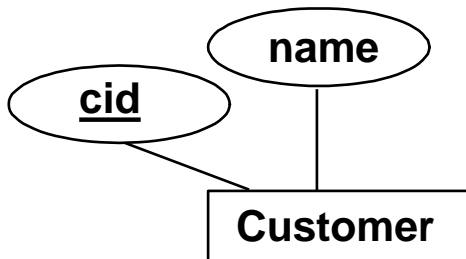
- **Binary 1-to-1 Relationships**
- **Participation constraints on both sides**
- (Note: From Step 1, we already have a relation created for **S** and one relation created for **T**.)
- Merge the relations for **S** and **T** into a single relation.



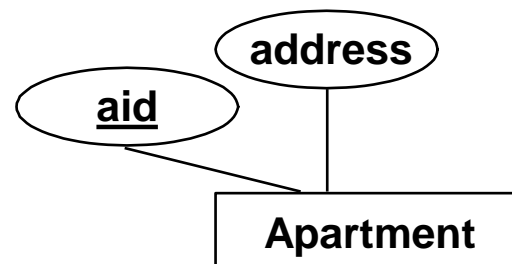


Step 3-B Examples

```
CREATE TABLE Customer  
(cid: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (cid))
```



```
CREATE TABLE Apartment  
(aid: INTEGER,  
address: CHAR(30),  
PRIMARY KEY (aid))
```

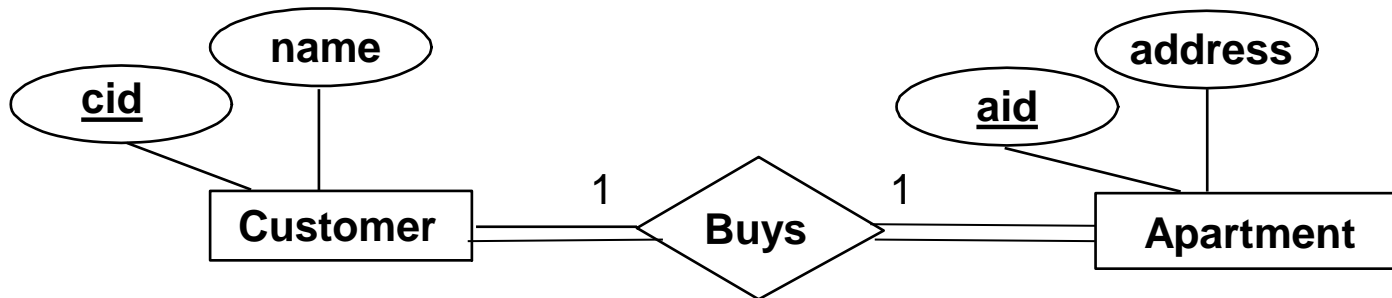




Step 3-B Examples

First, get rid of the Customer and Apartment relations.
Then, create the following relation:

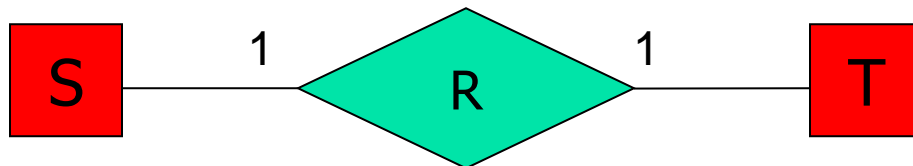
```
CREATE TABLE Customer_Apartment  
(cid: INTEGER,  
name: CHAR(30),  
aid: INTEGER,  
address: CHAR(30),  
PRIMARY KEY (cid))
```





Step 3-C

- **Binary 1-to-1 Relationships**
- **No participation constraints**
- (Note: From Step 1, we already have a relation created for **S** and one relation created for **T**.)
- Relations for **S** and **T** stay the way they are.
- Create a new relation for relationship **R**, use **S** and **T**'s primary keys as foreign keys in **R**.



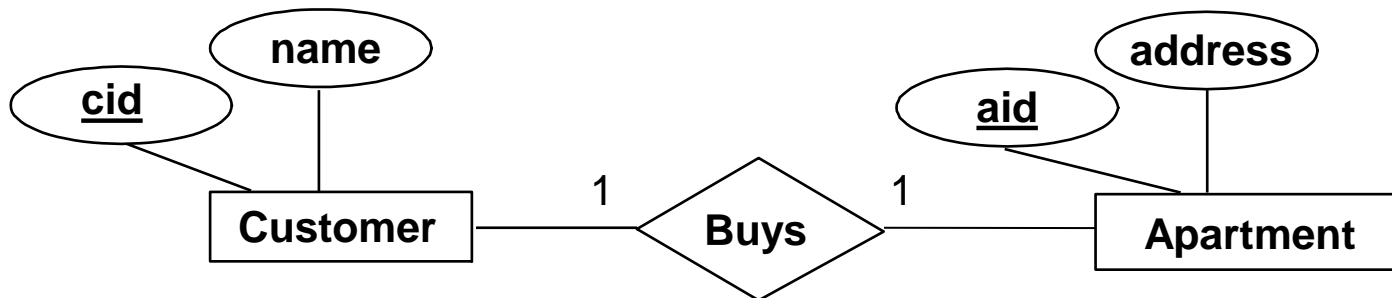


Step 3-C Examples

```
CREATE TABLE Customer  
(cid: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (cid))
```

```
CREATE TABLE Apartment  
(aid: INTEGER,  
address: CHAR(30),  
PRIMARY KEY (aid))
```

```
CREATE TABLE Buys  
(cid: INTEGER,  
aid: INTEGER,  
FOREIGN KEY cid REFERENCES Customer,  
FOREIGN KEY aid REFERENCES Apartment,  
PRIMARY KEY (cid))
```





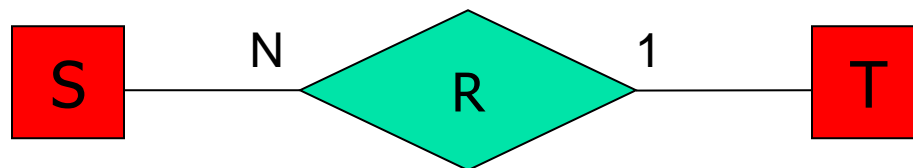
Conversion Algorithm

- We will follow a step-by-step algorithm.
- Step 1: Mapping of Regular Entities
 - Mapping of Subclass/Superclass
- Step 2: Mapping of Weak Entities
- Step 3: Mapping of Binary 1:1 Relationships
- Step 4: Mapping of Binary 1:N Relationships
- Step 5: Mapping of Binary M:N Relationships
- Step 6: Mapping of Multivalued Attributes
- Step 7: Mapping of N-ary Relationships (eg: ternary)



Step 4

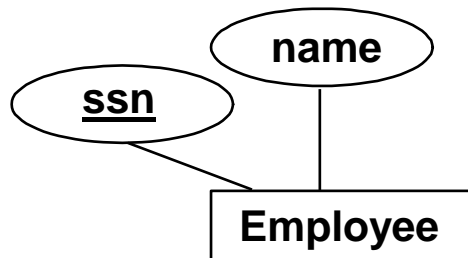
- **Binary 1-to-N Relationships (or N-to-1)**
- (Note: From Step 1, we already have a relation created for **S** and one relation created for **T**.)
- To handle relationship **R**:
 - Let **S** be the entity on the "N"-side of the relationship.
 - Add the primary key of **T** as foreign key in **S**.
 - Add the simple attributes of **R** to **S**.



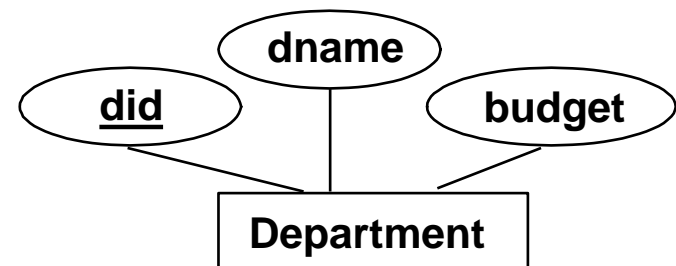


Step 4 Examples

```
CREATE TABLE Employee  
  (ssn: INTEGER,  
   name: CHAR(30),  
   PRIMARY KEY (ssn))
```



```
CREATE TABLE Department  
  (did: INTEGER,  
   dname: CHAR(30),  
   budget: REAL,  
   PRIMARY KEY (did))
```

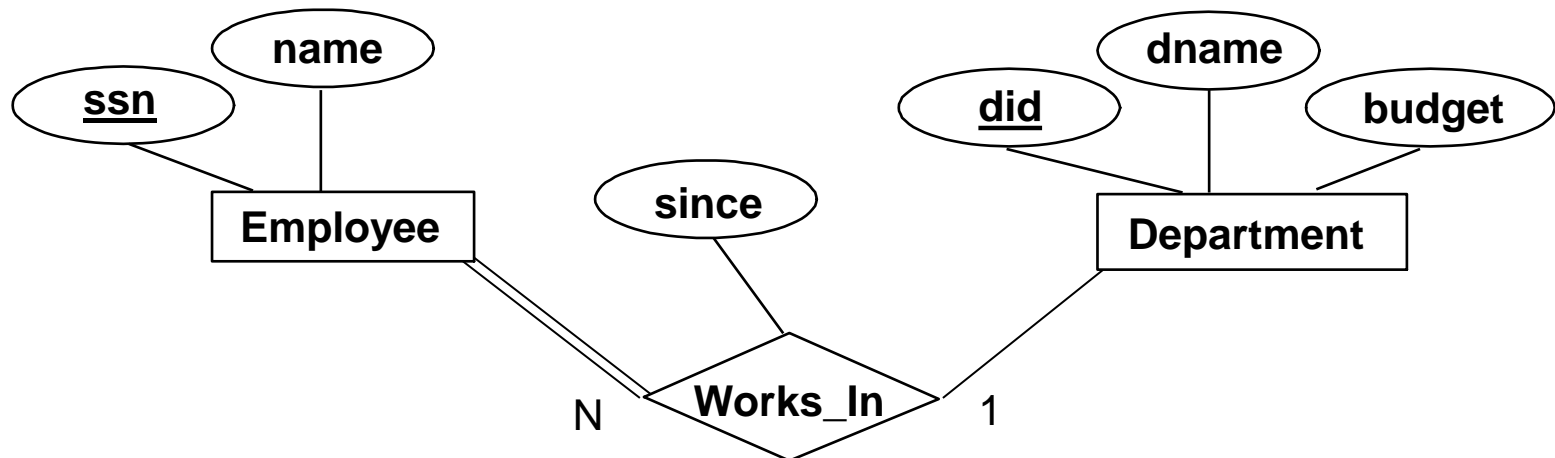




Step 4 Examples

```
CREATE TABLE Employee
(ssn: INTEGER,
name: CHAR(30),
PRIMARY KEY (ssn),
in_deptid: INTEGER,
FOREIGN KEY in_deptid
REFERENCES Department,
since: CHAR(10))
```

```
CREATE TABLE Department
(did: INTEGER,
dname: CHAR(30),
budget: REAL,
PRIMARY KEY (did))
```



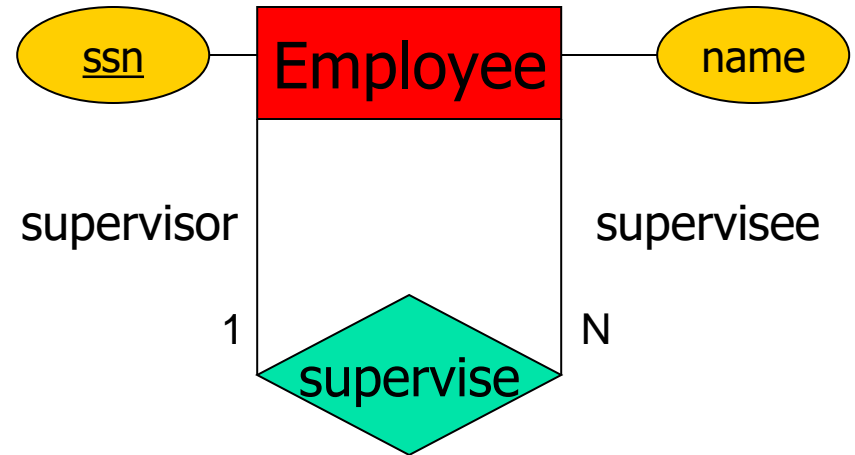


Step 4 Examples

```
CREATE TABLE Employee  
(ssn: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (ssn))
```



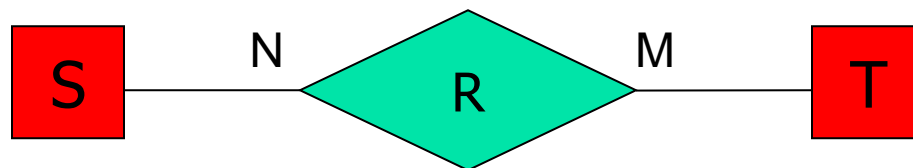
```
CREATE TABLE Employee  
(ssn: INTEGER,  
name: CHAR(30),  
PRIMARY KEY (ssn),  
supervisor_ssn: INTEGER,  
FOREIGN KEY supervisor_ssn  
REFERENCES Employee)
```





Step 5

- **Binary M-to-N Relationships**
- (Note: From Step 1, we already have a relation created for **S** and one relation created for **T**.)
- To handle the many-to-many relationship:
 - Create a new relation **R** for this relationship
 - Add **S** and **T**'s primary keys as foreign key in **R**, and add any attributes **R** may have
 - The combination of **S** and **T**'s primary keys is the primary key of **R**

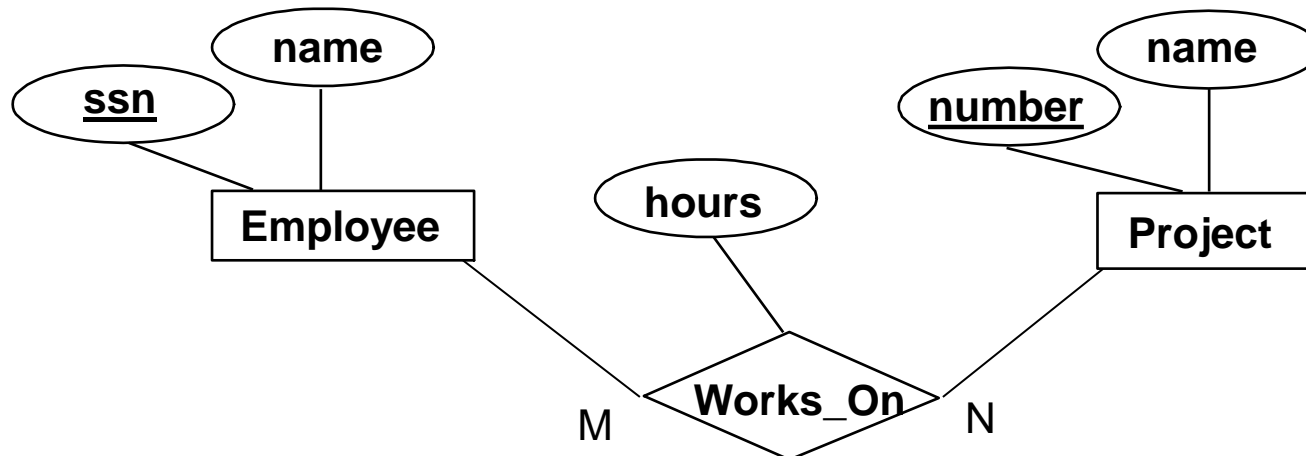




Step 5 Examples

```
CREATE TABLE Employee
(ssn: INTEGER,
name: CHAR(30),
PRIMARY KEY (ssn),
in_deptid: INTEGER,
FOREIGN KEY in_deptid
REFERENCES Department,
since: CHAR(10),
superv_ssn: INTEGER,
FOREIGN KEY superv_ssn
REFERENCES Employee)
```

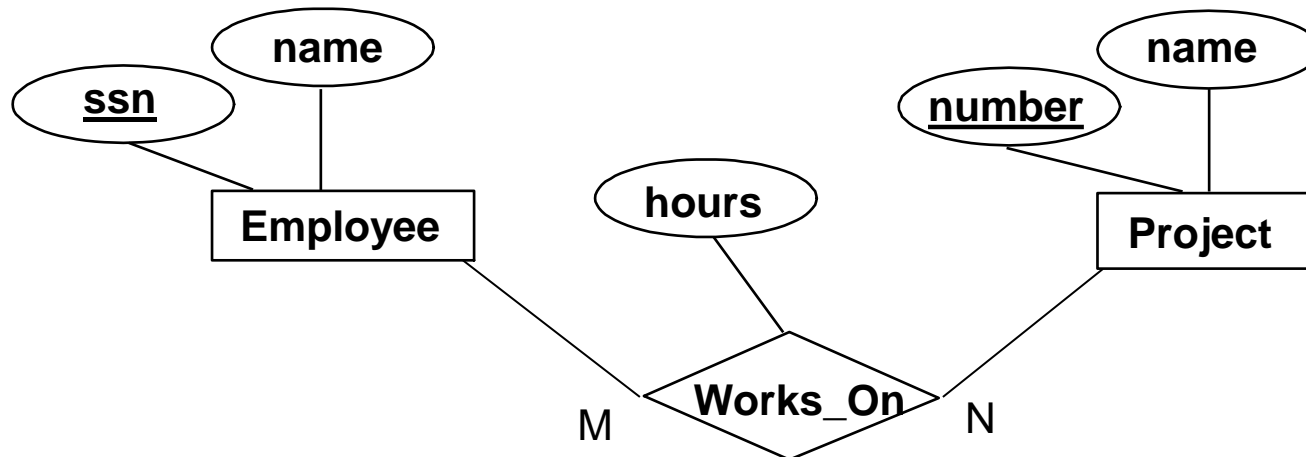
```
CREATE TABLE Project
(pnumber: INTEGER,
pname: CHAR(30),
PRIMARY KEY (pnumber))
```





Step 5 Examples

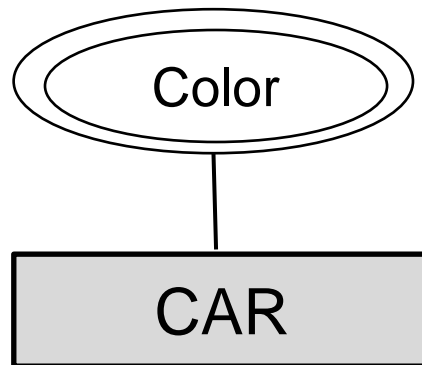
```
CREATE TABLE Works_On  
  (essn: INTEGER,  
   pno: INTEGER,  
   hours: INTEGER,  
   FOREIGN KEY essn  
             REFERENCES Employee,  
   FOREIGN KEY pno  
             REFERENCES Project,  
   PRIMARY KEY (essn, pno))
```





Step 6

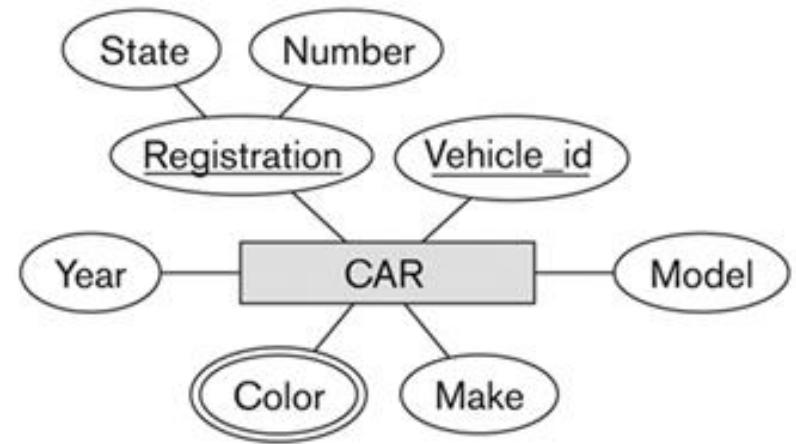
- **Multivalued Attributes**
- Create a new relation **R** for each multivalued attribute.
 - Add primary key of entity to **R**, as foreign key
 - Primary key of **R** is the combination of the multi-valued attribute + the added foreign key





Step 6 Examples

```
CREATE TABLE Car
(reg_state: CHAR(2),
 reg_number: CHAR(10),
 vehicle_id: CHAR(15),
 model: CHAR(10),
 make: CHAR(15),
 year: INTEGER,
 PRIMARY KEY (vehicle_id))
```

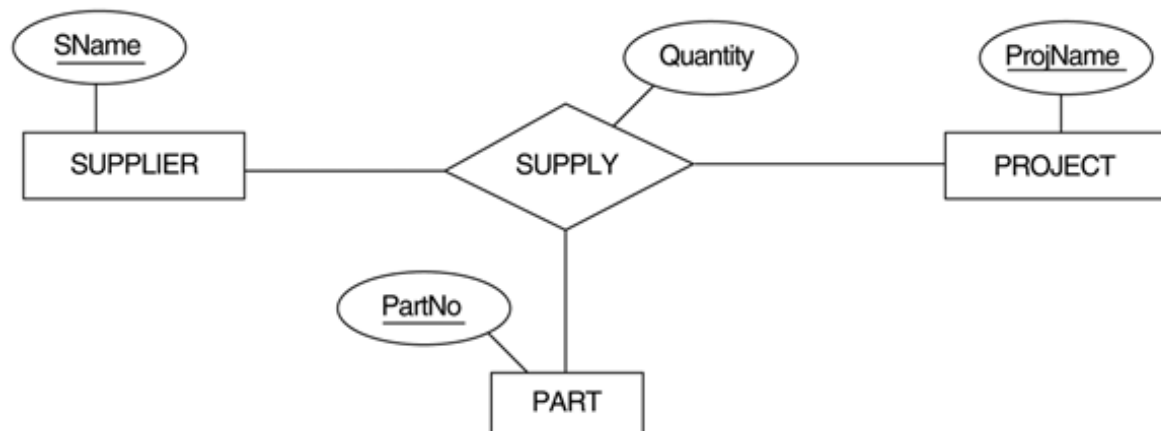


```
CREATE TABLE CarColors
(veh_id: CHAR(15),
 color: CHAR(10),
 PRIMARY KEY (veh_id, color),
 FOREIGN KEY veh_id REFERENCES Car)
```



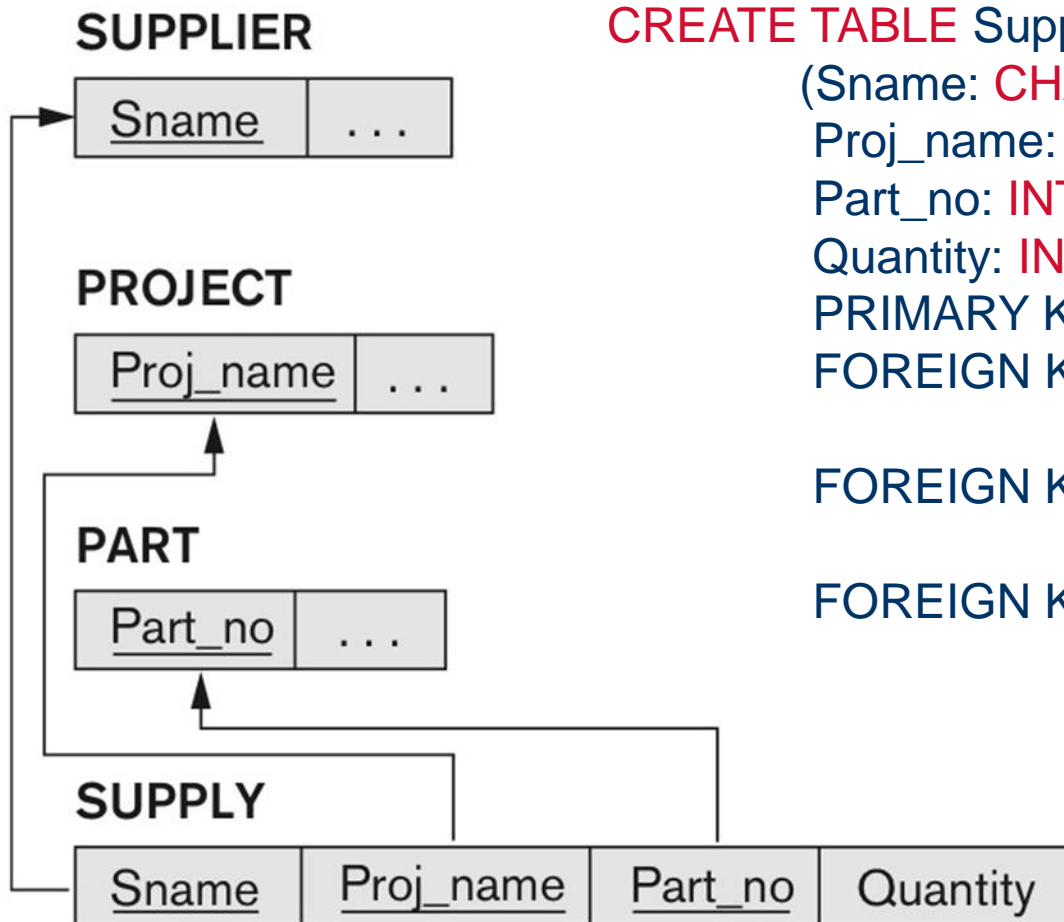
Step 7

- **N-ary Relationships (3-ary, 4-ary, etc.)**
- Create a new relation **R** to represent the relationship.
 - Add the primary keys of all participating entities to **R**, as foreign keys.
 - Add any attribute the relationship itself may have.
 - Primary key of **R** is the combination of all foreign keys added in the first step.





Step 7 Examples



```
CREATE TABLE Supply
(Sname: CHAR(20),
Proj_name: CHAR(10),
Part_no: INTEGER,
Quantity: INTEGER,
PRIMARY KEY (Sname, Proj_name, Part_no),
FOREIGN KEY Sname
REFERENCES Supplier,
FOREIGN KEY Proj_name
REFERENCES Project,
FOREIGN KEY Part_no
REFERENCES Part)
```



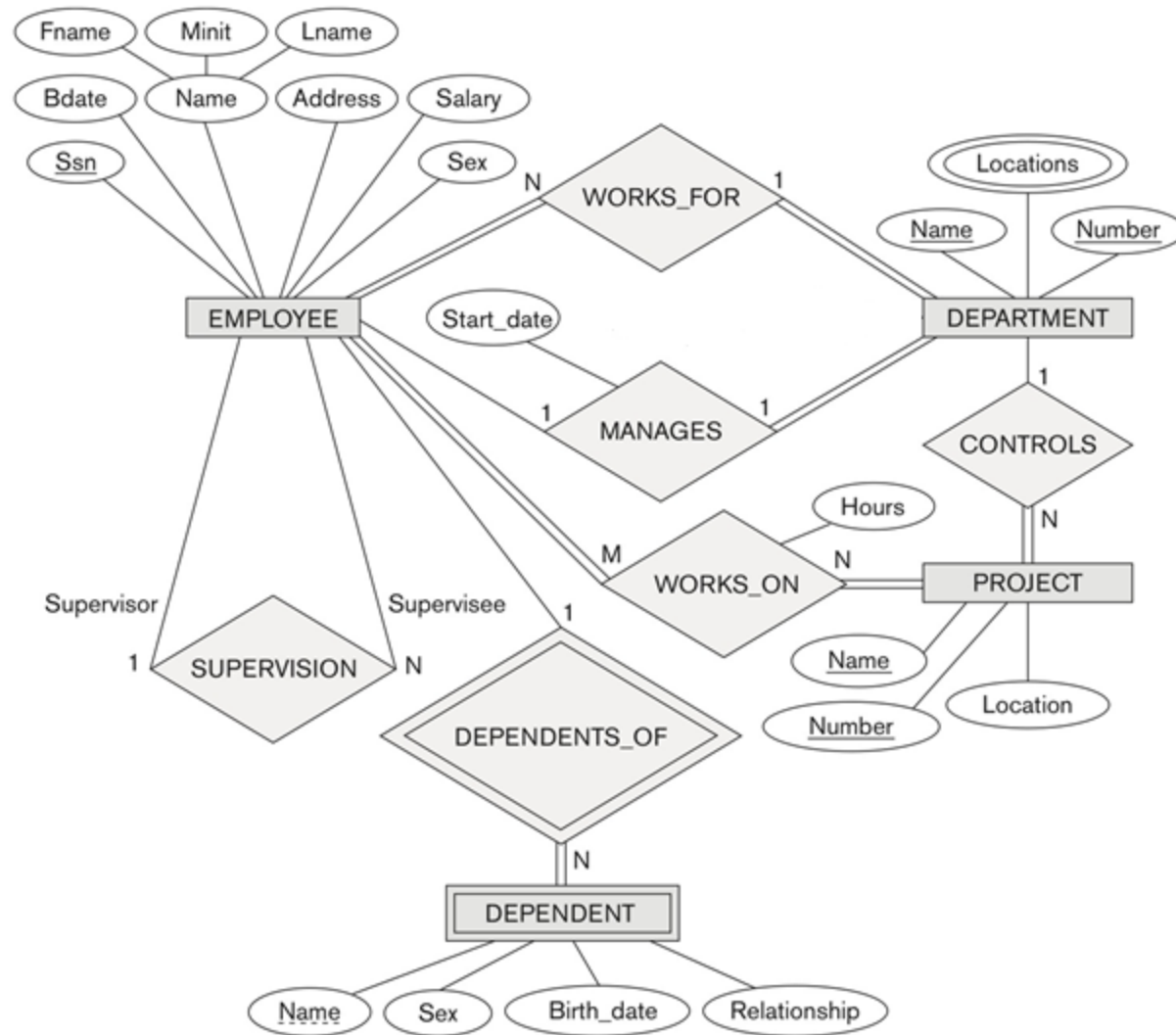
Done!

- Step 1: Mapping of Regular Entities
 - Mapping of Subclass/Superclass
- Step 2: Mapping of Weak Entities
- Step 3: Mapping of Binary 1:1 Relationships
- Step 4: Mapping of Binary 1:N Relationships
- Step 5: Mapping of Binary M:N Relationships
- Step 6: Mapping of Multivalued Attributes
- Step 7: Mapping of N-ary Relationships (eg: ternary)

- Final checks:
 - Check **all information in ER is captured** in the relational database (entity, relationship, attributes)
 - Check existence and correctness of **all primary keys**
 - Check existence and correctness of **all foreign keys**



Company Database





Step 1

- Handle EMPLOYEE, DEPARTMENT, PROJECT
 - Note: Locations of DEPARTMENT are not handled yet since it is multi-valued

```
CREATE TABLE Employee
(Ssn: INTEGER,
 Bdate: CHAR(10),
 Fname: CHAR(20),
 Minit: CHAR(1),
 Lname: CHAR(20),
 Address: CHAR(100),
 Salary: REAL,
 Sex: CHAR(10),
 PRIMARY KEY (Ssn))
```

```
CREATE TABLE Department
(Dname: CHAR(20),
 Dnumber: INTEGER,
 PRIMARY KEY (Dnumber))
```

```
CREATE TABLE Project
(Pname: CHAR(20),
 Pnumber: INTEGER,
 Plocation: CHAR(50),
 PRIMARY KEY (Pnumber))
```



Step 2

- Handle DEPENDENT, DEPENDENTS_OF

```
CREATE TABLE Emp_Dependents
(emp_ssn: INTEGER,
 dep_name: CHAR(30),
 dep_sex: CHAR(10),
 dep_birthdate: CHAR(10),
 dep_relationship: CHAR(10),
 PRIMARY KEY (emp_ssn, dep_name),
 FOREIGN KEY emp_ssn
 REFERENCES Employee)
```



Step 3

- Convert MANAGES relationship
 - Which option? → 3A
 - That means I must change the Department relation

```
CREATE TABLE Department  
(Dname: CHAR(20),  
Dnumber: INTEGER,  
PRIMARY KEY (Dnumber))
```



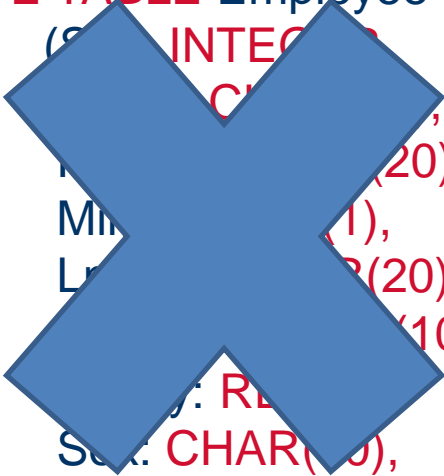
```
CREATE TABLE Department  
(Dname: CHAR(20),  
Dnumber: INTEGER,  
Manager_SSN: INTEGER,  
Manager_startdate: CHAR(10),  
PRIMARY KEY (Dnumber),  
FOREIGN KEY Manager_SSN REFERENCES Employee)
```



Step 4

- Handle WORKS_FOR, SUPERVISION, CONTROLS
 - I must change EMPLOYEE to handle WORKS_FOR

CREATE TABLE Employee
(Ssn: INTEGER,
Bdate: CHAR(10),
Fname: CHAR(20),
Minit: CHAR(1),
Lname: CHAR(20),
Address: CHAR(100),
Salary: REAL,
Sex: CHAR(10),
Works_In_Dno: INTEGER,
PRIMARY KEY (Ssn))

A large, thick blue 'X' mark is drawn over the SQL code on the left, indicating that this version of the table definition is incorrect or to be discarded.

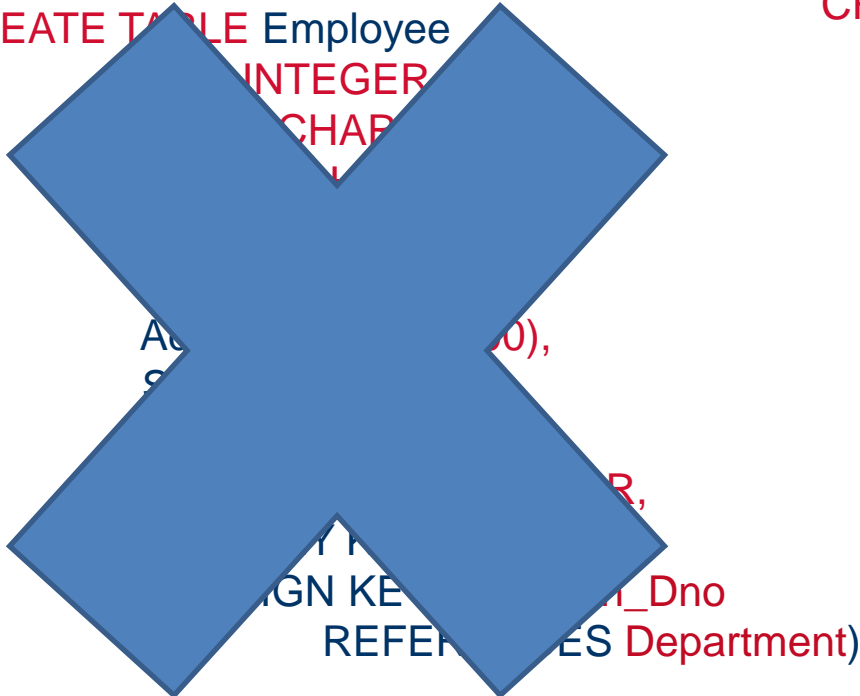
CREATE TABLE Employee
(Ssn: INTEGER,
Bdate: CHAR(10),
Fname: CHAR(20),
Minit: CHAR(1),
Lname: CHAR(20),
Address: CHAR(100),
Salary: REAL,
Sex: CHAR(10),
Works_In_Dno: INTEGER,
PRIMARY KEY (Ssn),
FOREIGN KEY Works_In_Dno
REFERENCES Department)



Step 4

- Handle WORKS_FOR, SUPERVISION, CONTROLS
 - I must change EMPLOYEE to handle WORKS_FOR
 - I must change EMPLOYEE to handle SUPERVISION

CREATE TABLE Employee
(Ssn: INTEGER,
Bdate: CHAR(10),
Fname: CHAR(20),
Minit: CHAR(1),
Lname: CHAR(20),
Address: CHAR(100),
Salary: REAL,
Sex: CHAR(10),
Works_In_Dno: INTEGER,
Supervisor_SSN: INTEGER,
PRIMARY KEY (Ssn),
FOREIGN KEY Works_In_Dno
REFERENCES Department)



CREATE TABLE Employee
(Ssn: INTEGER,
Bdate: CHAR(10),
Fname: CHAR(20),
Minit: CHAR(1),
Lname: CHAR(20),
Address: CHAR(100),
Salary: REAL,
Sex: CHAR(10),
Works_In_Dno: INTEGER,
Supervisor_SSN: INTEGER,
PRIMARY KEY (Ssn),
FOREIGN KEY Works_In_Dno
REFERENCES Department,
FOREIGN KEY Supervisor_SSN
REFERENCES Employee)



Step 4

- Handle WORKS_FOR, SUPERVISION, CONTROLS
 - I must change PROJECT to handle CONTROLS

CREATE TABLE Project
(Pname: CHAR(20),
Pnumber: INTEGER,
Plocation: CHAR(50),
PRIMARY KEY (Pnumber))

A large, semi-transparent blue 'X' mark is superimposed over the SQL code on the left, indicating that this version of the code is incorrect or being rejected.

CREATE TABLE Project
(Pname: CHAR(20),
Pnumber: INTEGER,
Plocation: CHAR(50),
Control_Dno: INTEGER,
PRIMARY KEY (Pnumber),
FOREIGN KEY Control_Dno
REFERENCES Department)



Step 5

- Handle WORKS_ON

```
CREATE TABLE Works_On
(Emp_Ssn: INTEGER,
 Proj_Pnum: INTEGER,
 Hours: INTEGER,
 PRIMARY KEY (Emp_Ssn, Proj_Pnum),
 FOREIGN KEY Emp_Ssn
      REFERENCES Employee,
 FOREIGN KEY Proj_Pnum
      REFERENCES Project)
```




Steps 6+7

- Step 6: Handle Locations attribute of DEPARTMENT

```
CREATE TABLE Dept_Locations  
  (Dnum: INTEGER,  
   Location: CHAR(30),  
   PRIMARY KEY (Dnum, Location),  
   FOREIGN KEY Dnum  
     REFERENCES Department)
```

- Step 7: No action



Final Result

- Dept_Locations (Dnum, Location)
- Works_On (Emp_Ssn, Proj_Pnum, Hours)
- Employee (Ssn, Bdate, Fname, Minit, Lname, Address, Salary, Sex, Works_In_Dno, Supervisor_Ssn)
- Department (Dname, Dnumber, Manager_Ssn, Manager_startdate)
- Emp_Dependents (Emp_ssn, Dep_name, Dep_sex, Dep_Birthdate, Dep_Relationship)
- Project (Pname, Pnumber, Plocation, Control_Dno)



Final Result

EMPLOYEE

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

DEPARTMENT

Dname	<u>Dnumber</u>	Mgr_ssn	Mgr_start_date
-------	----------------	---------	----------------

DEPT_LOCATIONS

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

PROJECT

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

WORKS_ON

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

DEPENDENT

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

