BIRZEIT UNIVERSITY

Faculty of Engineering and Tecnology

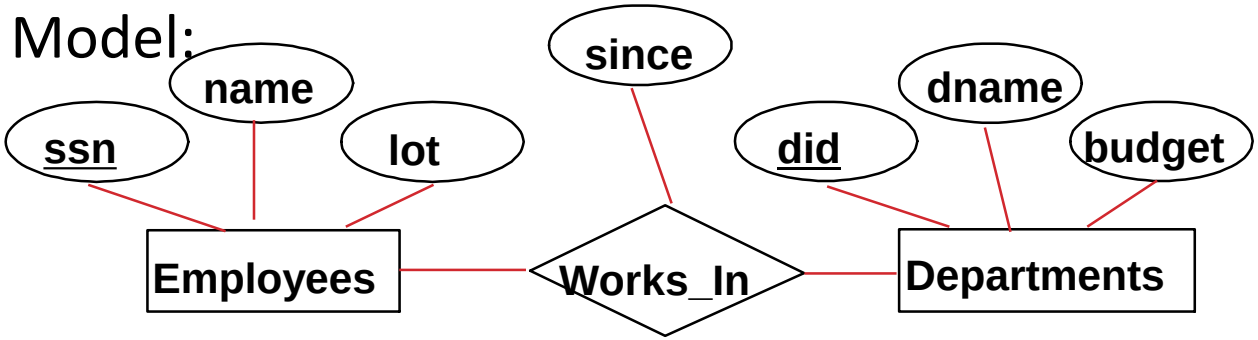
Computer Science Department

Relational Model

Chapter 3

Review

- E/R Model:



- Entities, relationships, attributes

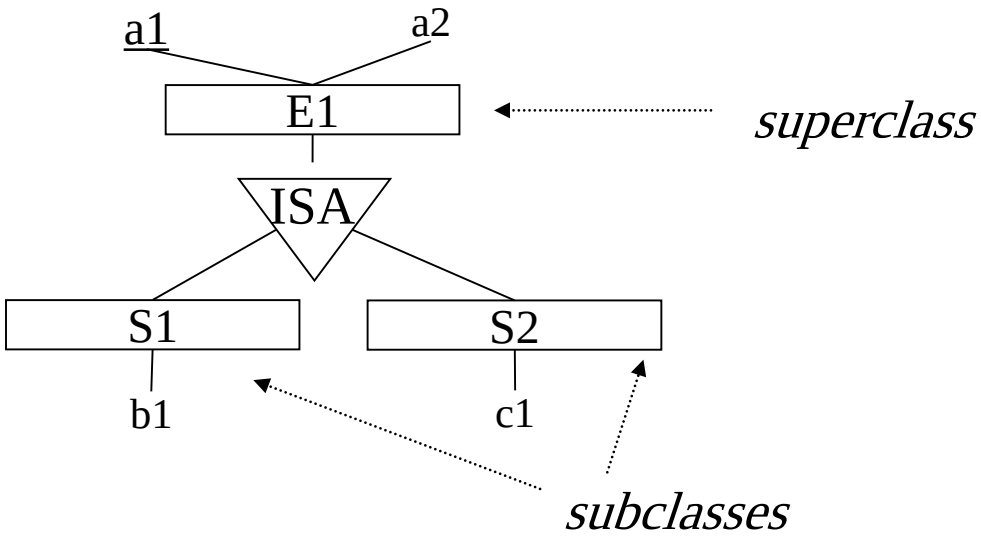
- Cardinalities: 1:1, 1:n, m:1, m:n
- Keys: superkeys, candidate keys, primary keys

Review

- Weak Entity sets, identifying relationship
- Discriminator, total participation, one-to-many

Review

- Generalization-specialization



Review

- Data models: framework for organizing and interpreting data

- E/R Model
- OO, Object relational, XML
- Relational Model
  - Intro
  - E/R to relational
  - SQL preview

Relational Data Model

- Introduced by Ted Codd (early 70') (Turing Award, '81)

- Relational data model contributes:
  1. Separation of logical and physical data models (data independence)
  2. Declarative query languages
  3. Formal semantics
  4. Query optimization (key to commercial success)

Relations

account =

bname	acct_no	balance
Downtown	A-101	500
Brighton	A-202	450
Brookline	A312	600

Rows (tuples, records)
Columns (attributes)
Tables (relations)

Why relations?

Relations

- Mathematical relations (from set theory):

Given 2 sets R={ 1, 2, 3, 5}, S={3, 4}

- R x S = {(1,3), (1, 4), (2, 3), (2,4), (3,3), (3,4), (5,3), (5,4)}
- A relation between R and S is any subset of R x S

  e.g., {(1,3), (2,4), 5,3)}

- Database relations:

Given attribute domains:

bname = {Downtown, Brighton, ….}

acct_no = { A-101, A-102, A-203, …}

balance = { …, 400, 500, …}

{ (Downtown, A-101, 500),
(Brighton, A-202, 450),
(Brookline, A-312, 600)}

account *subset of* bname x acct_no x balance

Storing Data in a Table

sid	name	major	age	gpa
53666	Duaa	CE	18	3.4
53688	Ali	CE	18	3.2
53650	Mohammad	CS	19	3.8

- Data about individual students
- One row per student
- How to represent course enrollment?

Storing More Data in Tables

- Students may enroll in more that one course
- Most efficient: keep enrollment in separate table

Enrolled

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

Linking Data from Multiple Tables

- How to connect student data to enrollment?
- Need a *Key*

Enrolled

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

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

Relational Data Model: Formal Definitions

- *Relational database:* a set of *relations*.
- *Relation:* made up of 2 parts:
  - *Instance* : a *table*, with rows and columns.
    - #rows = *cardinality*
  - *Schema* : specifies name of relation, plus name and type of each column.
    - E.g. Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)
    - #fields = *degree / arity*
- Can think of a relation as a *set* of rows or *tuples*.
  - i.e., all rows are distinct

In other words…

- Data Model – a way to organize information
- Schema – one particular organization,
  - i.e., a set of fields/columns, each of a given type
- Relation
  - a name
  - a schema
  - a set of tuples/rows, each following organization specified in schema

Example Instance of Students Relation

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

Cardinality = 3, arity (degree) = 5 , all rows distinct

SQL - A language for Relational DBs

- SQL: Structured Query language

  - Data Definition Language (DDL)
    - create, modify, delete relations
    - specify constraints
    - administer users, security, etc.

  - Data Manipulation Language (DML)
    - Specify *queries* to find tuples that satisfy criteria
    - add, modify, remove tuples

SQL Overview

- CREATE TABLE <name> (<field> <domain>, …)
- INSERT INTO <name> (<field names>)
  VALUES (<field values>)
- DELETE FROM <name>
  WHERE <condition>
- UPDATE <name>
  SET <field name> = <value>
  WHERE <condition>
- SELECT <fields>
  FROM <name>
  WHERE <condition>

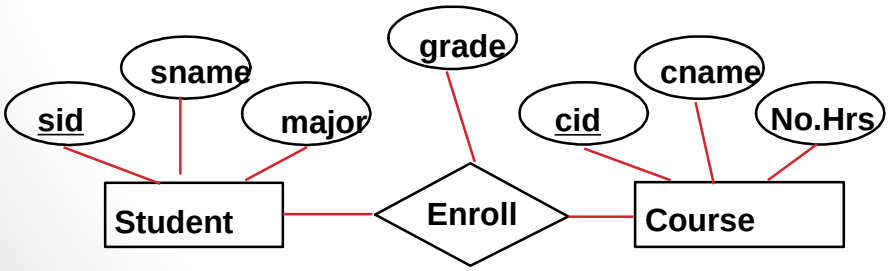
Creating Relations in SQL

- Creates the Students relation.

```
CREATE TABLE Students
      (sid CHAR(9),
       name CHAR(20),
       major CHAR(10),
       age INTEGER,
       gpa REAL)
```

- Note: the type (domain) of each field is specified, and enforced by the DBMS
  - whenever tuples are added or modified.

- Another example: the Enrolled table holds information about courses students take.




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

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO  Students (sid, name, major, age, gpa)
     VALUES  ('53688', 'Alaa', 'CE', 18, 83.4)
```

Can delete all tuples satisfying some condition (e.g., name = Smith):

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

- **Powerful variants of these commands are available; more later!**

Keys

- Integrity Constraints (IC): conditions that restrict the data that can be stored in the database
- Keys are a way to associate tuples in different relations
- Keys are one form of integrity constraint (IC)

Enrolled

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

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

Primary Keys - Definitions

- Key: A minimal set of attributes that uniquely identify a tuple
- A set of fields is a *superkey* if:
  - No two distinct tuples can have same values in all key fields
- A set of fields is a candidate *key* for a relation if :
  - It is a superkey
  - No subset of the fields is a superkey
- >1 candidate keys for a relation?
  - one of the keys is chosen (by DBA) to be the *primary key*.
- E.g.
  - *sid* is a key for Students.
  - What about *name*?
  - The set {*sid, gpa*} is a superkey.

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.

- "For a given student and course, there is a single grade."
- VS.
- "Students can take only one course, and receive a single grade for that course; further, no two students in a course receive the same grade."

```
CREATE TABLE Enrolled
   (sid CHAR(20)
     cid  CHAR(20),
     grade integer,
     PRIMARY KEY (sid,cid))
```

```
CREATE TABLE Enrolled
   (sid CHAR(20)
     cid  CHAR(20),
     grade integer,
     PRIMARY KEY  (sid),
     UNIQUE (cid, grade))
```

Foreign Keys

- A Foreign Key is a field whose values are keys in another relation.

Enrolled

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

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

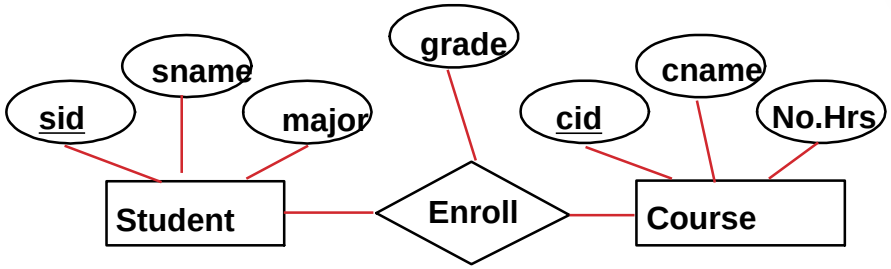
Foreign Keys, Referential Integrity

- ***Foreign key*** **: Set of fields in one relation used to `refer' to tuples in another relation.**
  - Must correspond to primary key of the second relation.
  - Like a `logical pointer'.

- **E.g. *sid* in Enrolled is a foreign key referring to Students:**
  - Enrolled(*sid*: string, *cid*: string, *grade*: string)
  - If all foreign key constraints are enforced, *referential integrity* is achieved (i.e., no dangling references.)

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.




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

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

Integrity Constraints (ICs)

- **IC**: condition that must be true for *any* instance of the database;
  - e.g., *domain constraints*.
  - ICs are specified when schema is defined.
  - ICs are checked when relations are modified.

- A *legal* instance of a relation is one that satisfies all specified ICs.
  - DBMS should not allow illegal instances.

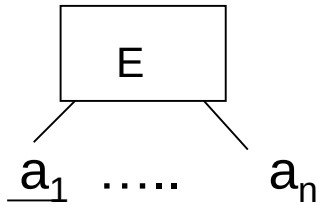
- If the DBMS checks ICs, stored data is more faithful to real-world meaning.
  - Avoids data entry errors, too!

E/R to Relations

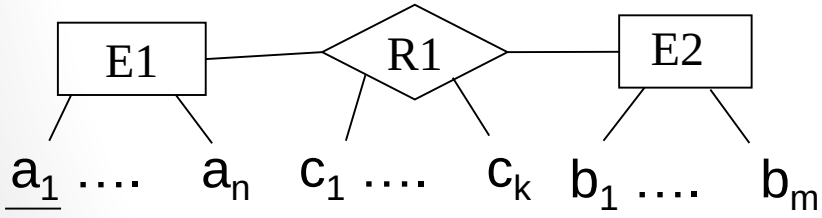
E/R diagram

Relational schema, e.g.

account=(bname, acct_no, bal)



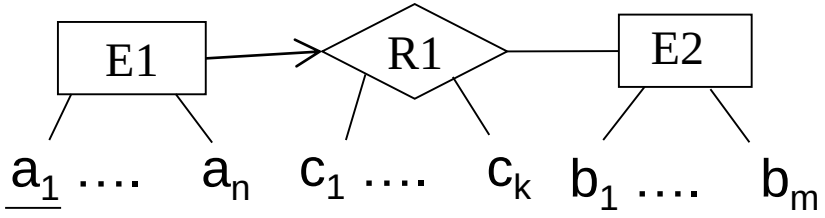
$E = (\underline{a_1}, \ \ldots, \ a_n)$



$R1 = (\underline{a_1, b_1}, c_1, \ \ldots, \ c_k)$

More on relationships

- What about:



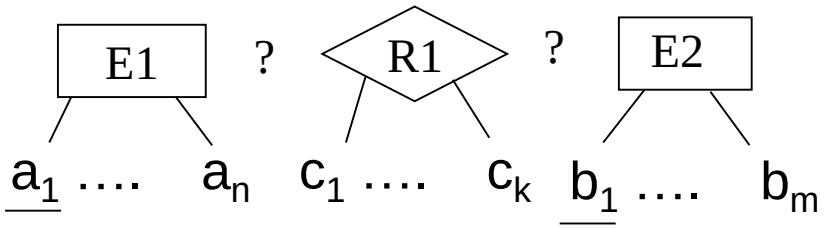
$$R1 = (\underline{a_1}, b_1, c_1, \ldots, c_k)$$

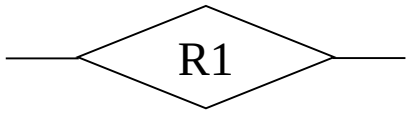
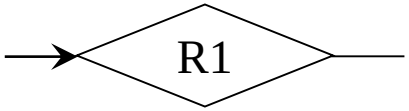
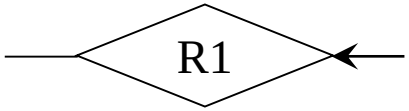
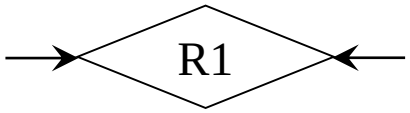
- Could have :

  since a_1 is the key for R1 (also for E1=($\underline{a_1}$, …., a_n))

- Another option is to merge E1 and R1
  - ignore R1
  - Add b1, c1, …., ck to E1 instead, i.e.
  - E1=($\underline{a_1}$, …., a_n, b_1, c_1, …, c_k)

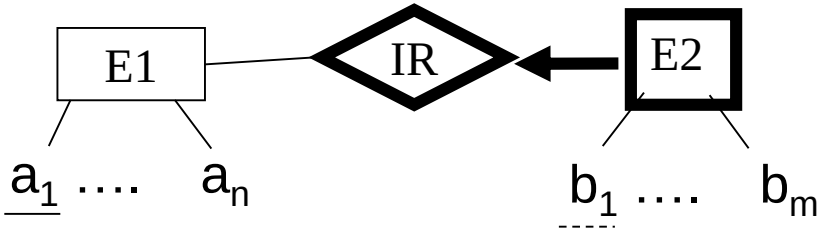
Any problem?



	E1 = ($\underline{a_1}$, …, a_n) E2 = ($\underline{b_1}$, …, b_m) R1 = ($\underline{a_1}$, $\underline{b_1}$, c_1 …, c_k)
	E1 = ($\underline{a_1}$, …, a_n , b_1, c_1, …, c_k) E2 = ($\underline{b_1}$, …, b_m)
	E1 = ($\underline{a_1}$, …, a_n) E2 = ($\underline{b_1}$, …, b_m , a_1, c_1, …, c_k)
	Treat as n:1 or 1:m

E/R to Relational

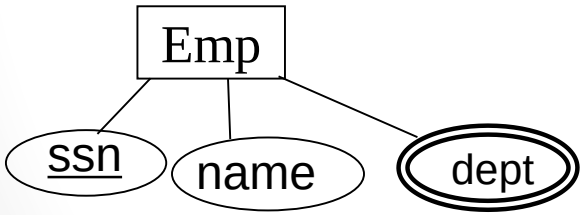
- Weak entity sets



E1 = ($\underline{a_1}$, …, a_n)

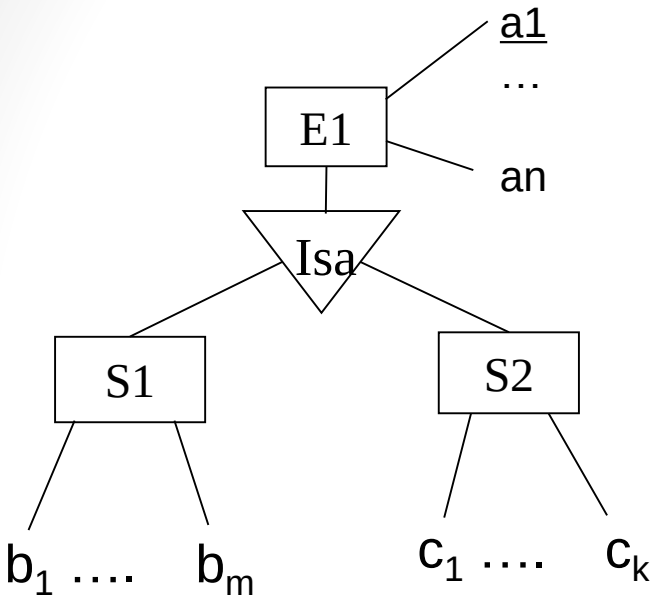
E2 = ($\underline{a_1}$, $\underline{b_1}$, …, b_m)

- Multivalued Attributes



Emp = (ssn, name)
Emp-Dept = (ssn, dept)

E/R to Relational



Method 1:

$E = (\underline{a_1}, \ldots, a_n)$

$S1 = (\underline{a_1}, b_1, \ldots, b_m)$

$S2 = (\underline{a_1}, c_1 \ldots, c_k)$

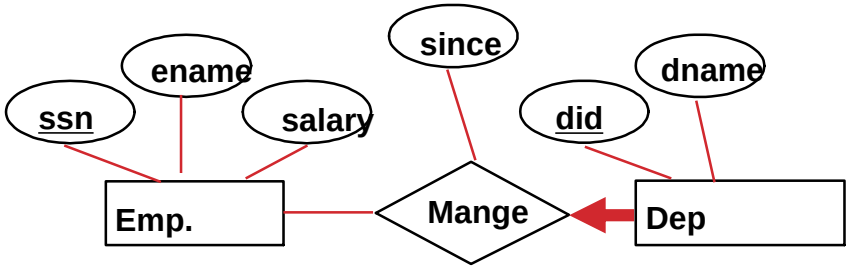
Method 2:

$S1 = (\underline{a_1}, \ldots, a_n, b_1, \ldots, b_m)$

$S2 = (\underline{a_1}, \ldots, a_n, c_1 \ldots, c_k)$

Q: When is method 2 not possible?

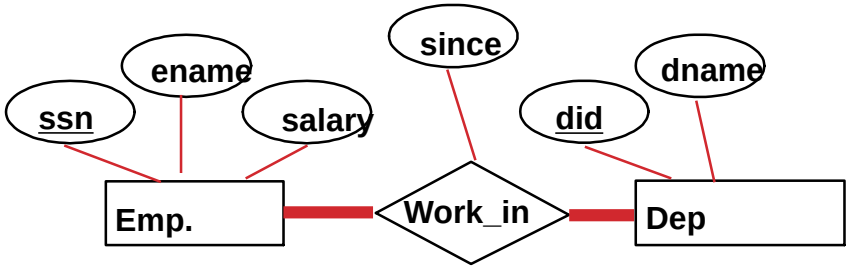
Relationships with Participation constraint




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

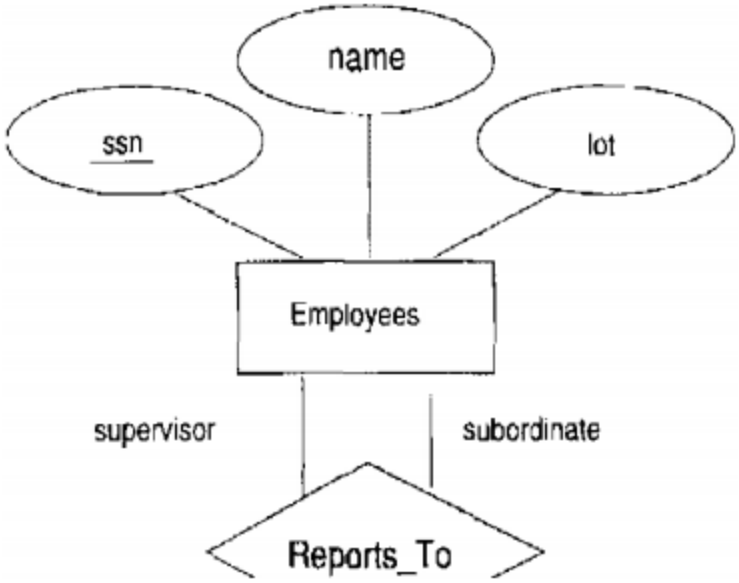
ON DELETE NO ACTION)

Relationships with Participation constraint



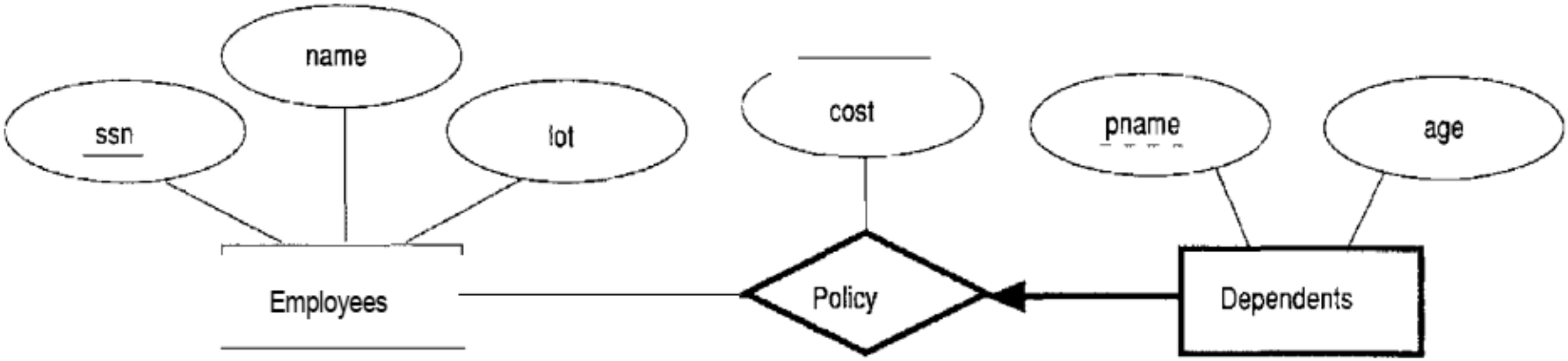
CREATE TABLE work_in (
di
ss
si
PF
FC N
FC N(
)

assertion




```
CREATE TABLE Reports_To (
        supervisor...ssn    CHAR(11),
        subordinate...ssn   CHAR(11),
        PRIMARY KEY (supervisor_ssn, subordinate_ssn),
        FOREIGN KEY (supervisor...ssn) REFERENCES Employees(ssn),
        FOREIGN KEY (subordinate...ssn) REFERENCES Employees(ssn) )
```

Weak entity




```
CREATE TABLE Dep_Policy (pname    CHAR(20),
                         age      INTEGER,
                         cost     REAL,
                         ssn      CHAR(11),
                         PRIMARY KEY (pname, ssn),
                         FOREIGN KEY (ssn) REFERENCES Employees
                                 ON DELETE CASCADE )
```

Translate the following ER to RM using create table statements