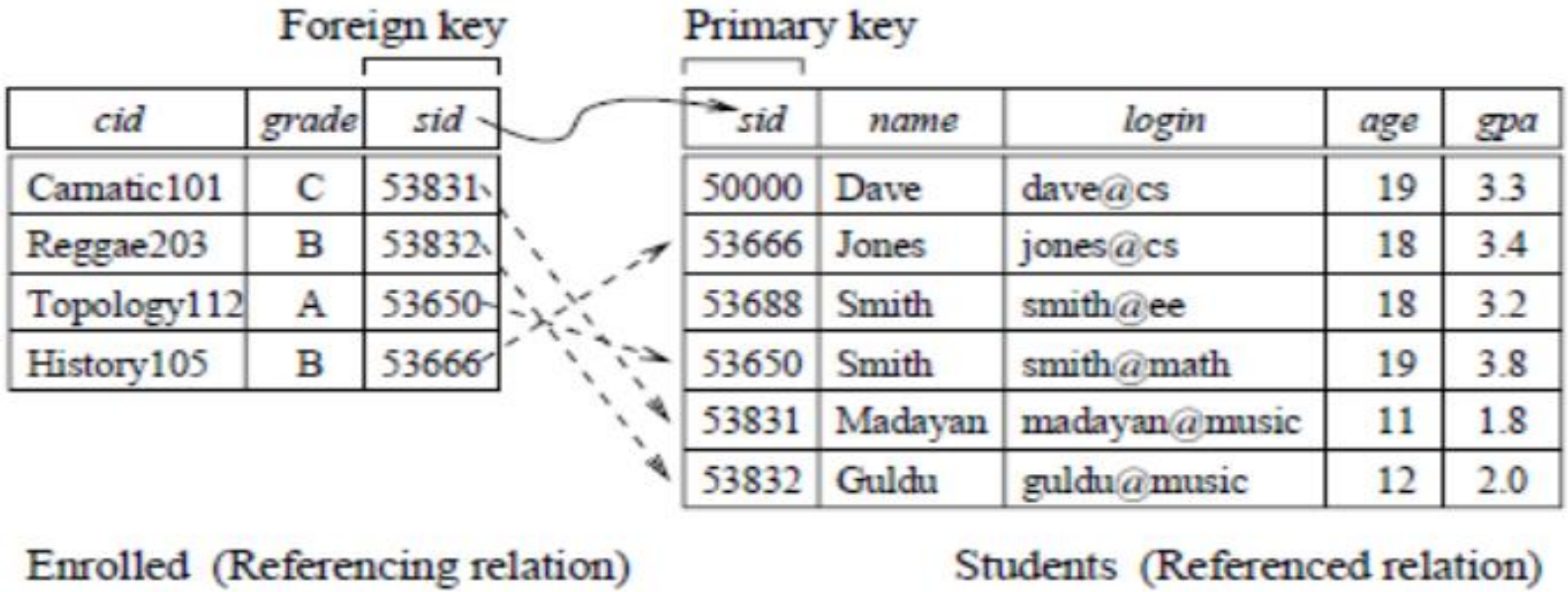
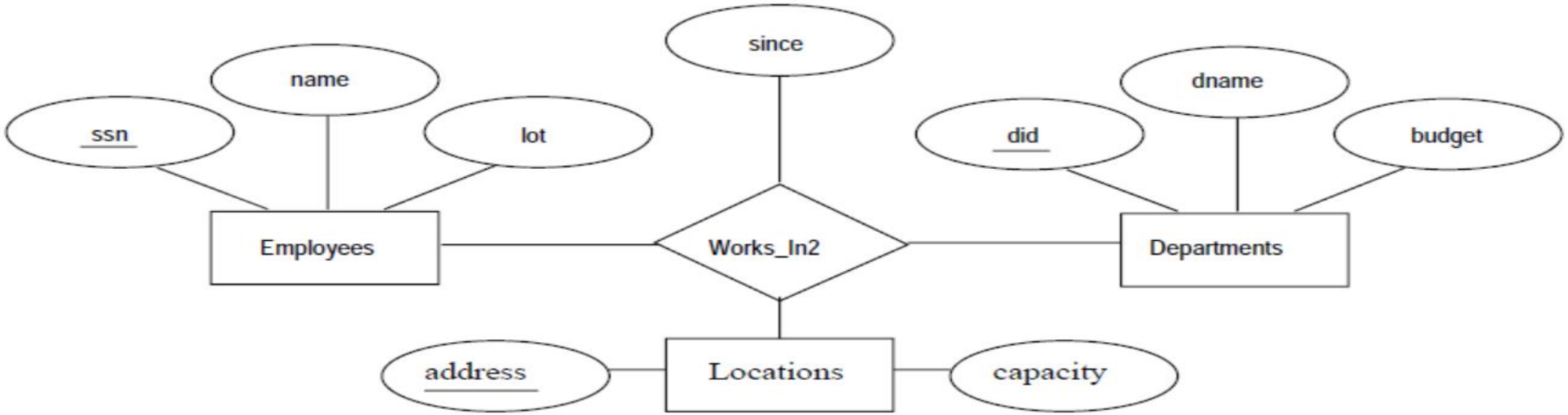
FOREIGN KEY/REFERENTIAL INTEGRITY CONSTRAINT

- Foreign Key is a column(s) that references a column(s) of a table, it can be the same table also
- It rejects an INSERT or UPDATE of a value, if a corresponding value does not currently exist in the master table
- Parent that is being referenced has to be unique or Primary key
- Child can have duplicates and nulls, unless specified explicitly
- Parent record can be deleted only when no corresponding child records exist in the child table
- Master table cannot be updated if child record exists
- If the ON DELETE CASCADE option is set, a DELETE operation in the master table will trigger a DELETE operation for corresponding records in all the child tables
- If the ON DELETE SET NULL option is set, a DELETE operation in the master table will set the value held by the foreign key of the child table to NULL

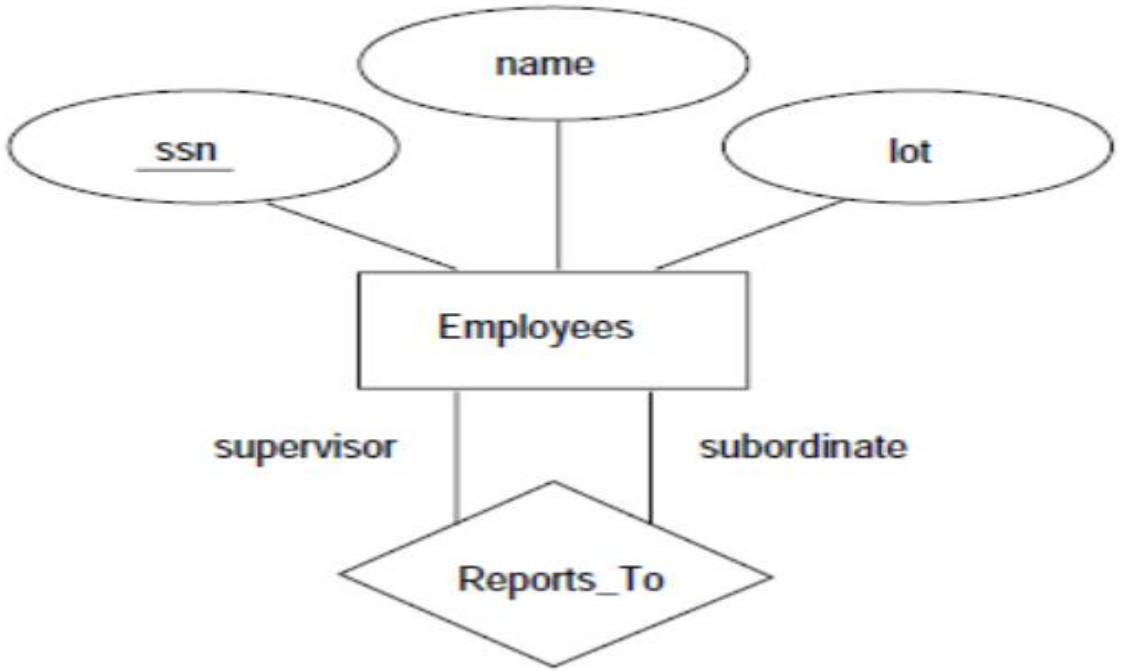

Foreign key
Primary key
cid
grade
sid
Carnatic101
C
53831
Reggae203
B
53832
Topology112
A
53650
History105
B
53666
sid
name
login
age
gpa
50000
Dave
dave@cs
19
3.3
53666
Jones
jones@cs
18
3.4
53688
Smith
smith@ee
18
3.2
53650
Smith
smith@math
19
3.8
53831
Madayan
madayan@music
11
1.8
53832
Guldu
guldu@music
12
2.0
Enrolled (Referencing relation)
Students (Referenced relation)

Ternary Relationship sets




```
CREATE TABLE Works_In2 ( ssn       CHAR(11),
                         did       INTEGER,
                         address   CHAR(20),
                         since     DATE,
                         PRIMARY KEY (ssn, did, address),
                         FOREIGN KEY (ssn) REFERENCES Employees,
                         FOREIGN KEY (address) REFERENCES Locations,
                         FOREIGN KEY (did) REFERENCES Departments )
```

Reports_to Relationship set

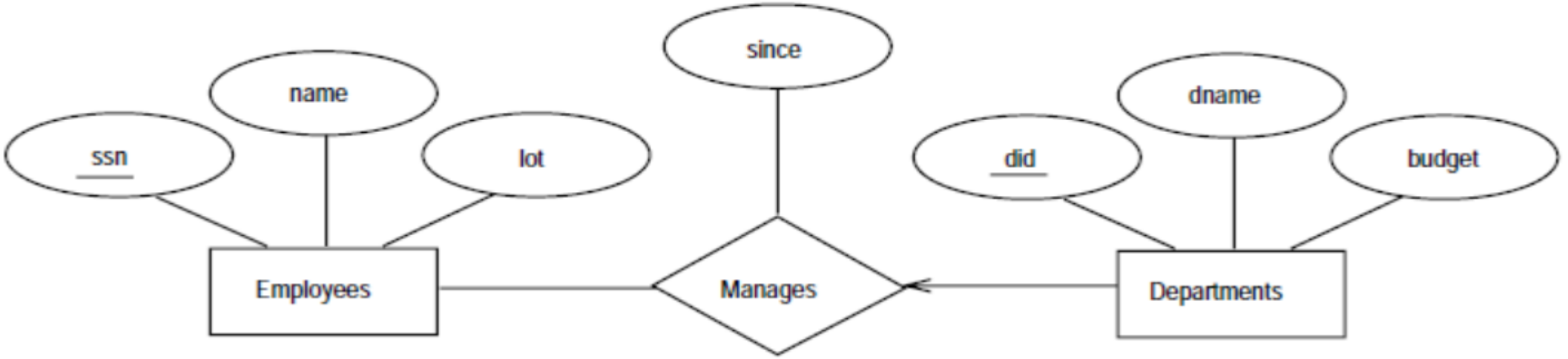



```
CREATE TABLE Reports_To (
        supervisor_ssn   CHAR(11),
        subordinate_ssn  CHAR(11),
        PRIMARY KEY (supervisor_ssn, subordinate_ssn),
        FOREIGN KEY (supervisor_ssn) REFERENCES Employees(ssn),
        FOREIGN KEY (subordinate_ssn) REFERENCES Employees(ssn) )
```

```
CREATE TABLE Dept_Mgr ( did       INTEGER,
                        dname     CHAR(20),
                        budget    REAL,
                        ssn       CHAR(11),
                        since     DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY (ssn) REFERENCES Employees )
```

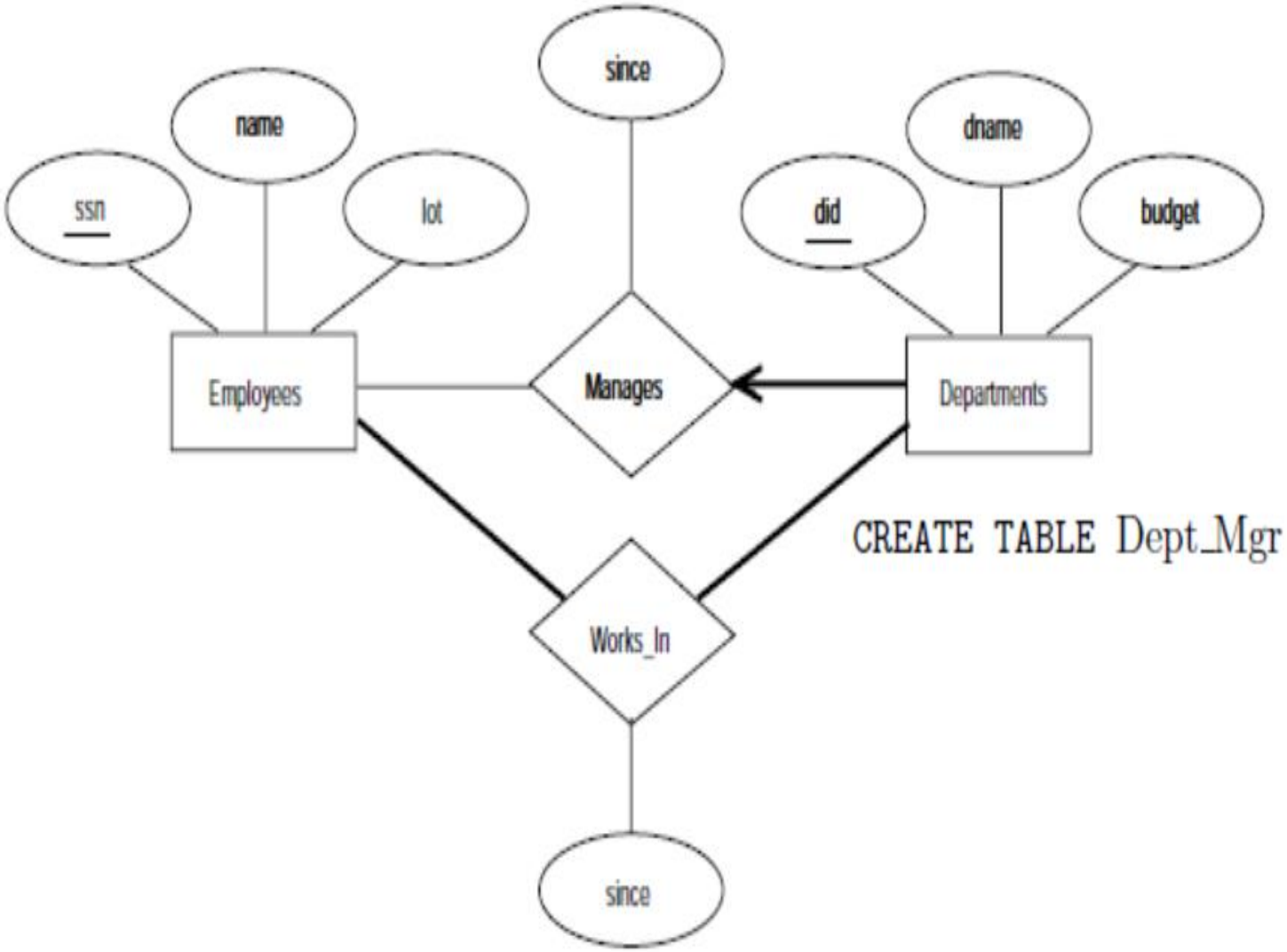
Note that *ssn* can take on *null* values.

Translating relationship sets with constraints



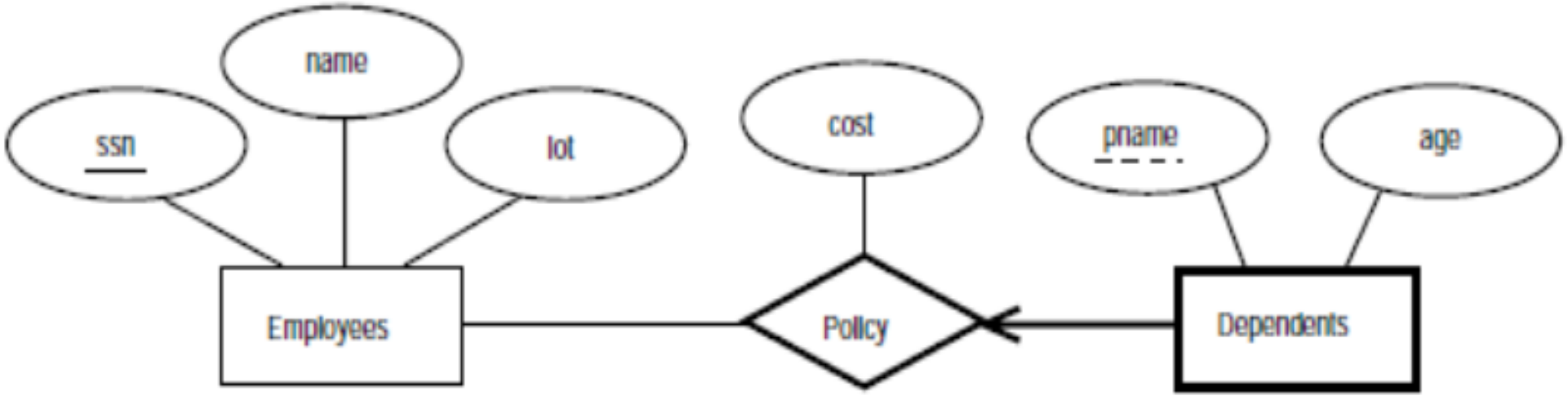

```
CREATE TABLE Manages (  ssn       CHAR(11),
                        did       INTEGER,
                        since     DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY (ssn) REFERENCES Employees,
                        FOREIGN KEY (did) REFERENCES Departments )
```

Translating relationship sets with participation constraints



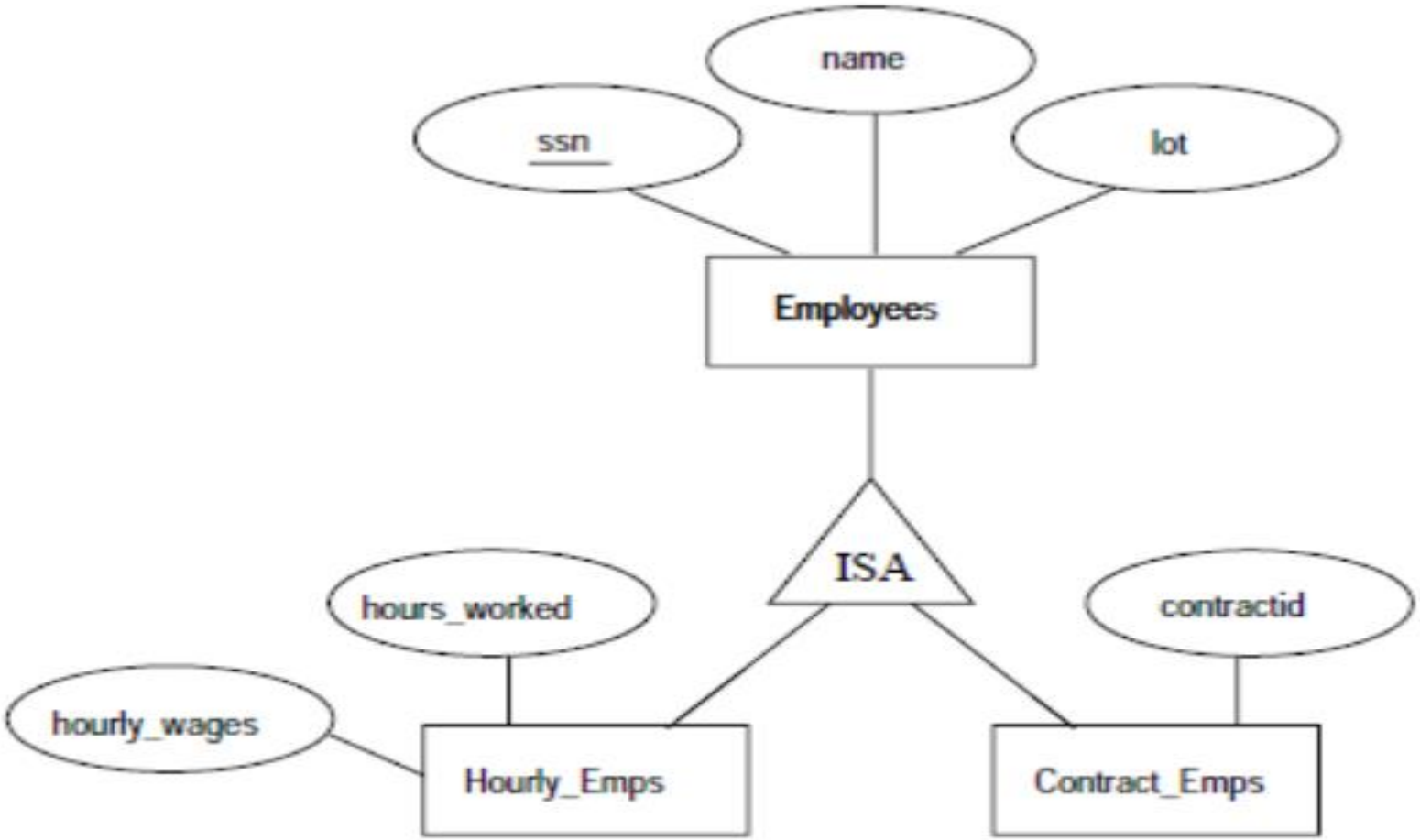

```
CREATE TABLE Dept_Mgr ( did       INTEGER,
                        dname     CHAR(20),
                        budget    REAL,
                        ssn       CHAR(11) NOT NULL,
                        since     DATE,
                        PRIMARY KEY (did),
                        FOREIGN KEY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

Translating Weak Entity Sets




```
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);
```

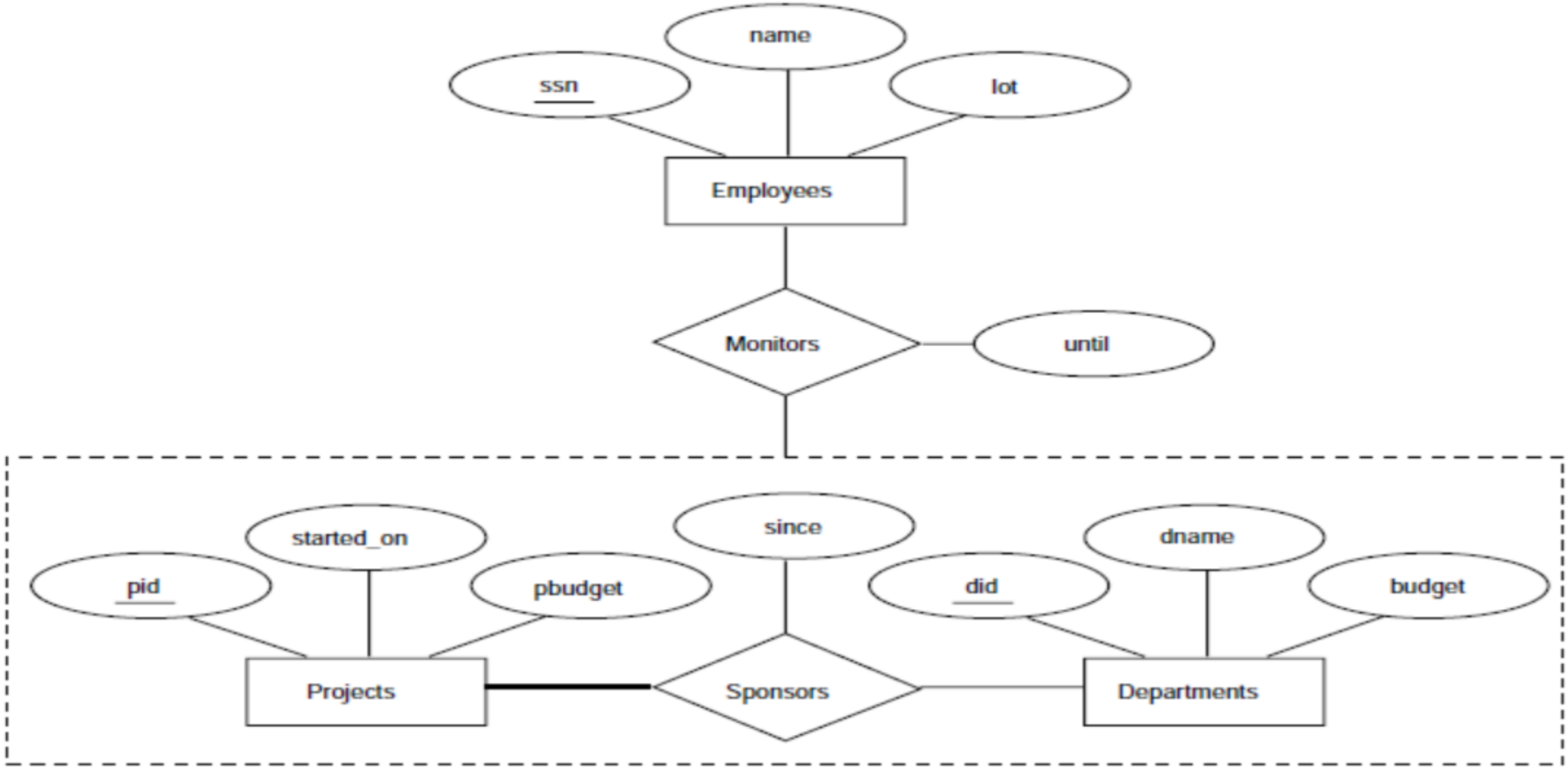
Translating Class Hierarchies



1. We can map each of the entity sets Employees, Hourly_Emps, and Contract_Emps to a distinct relation. The Employees relation is created as in Section 2.2. We discuss Hourly_Emps here; Contract_Emps is handled similarly. The relation for Hourly_Emps includes the *hourly_wages* and *hours_worked* attributes of Hourly_Emps. It also contains the key attributes of the superclass (*ssn*, in this example), which serve as the primary key for Hourly_Emps, as well as a foreign key referencing the superclass (Employees). For each Hourly_Emps entity, the value of the *name* and *lot* attributes are stored in the corresponding row of the superclass (Employees). Note that if the superclass tuple is deleted, the delete must be cascaded to Hourly_Emps.

2. Alternatively, we can create just two relations, corresponding to Hourly_Emps and Contract_Emps. The relation for Hourly_Emps includes all the attributes of Hourly_Emps as well as all the attributes of Employees (i.e., *ssn*, *name*, *lot*, *hourly_wages*, *hours_worked*).

Translating Aggregations



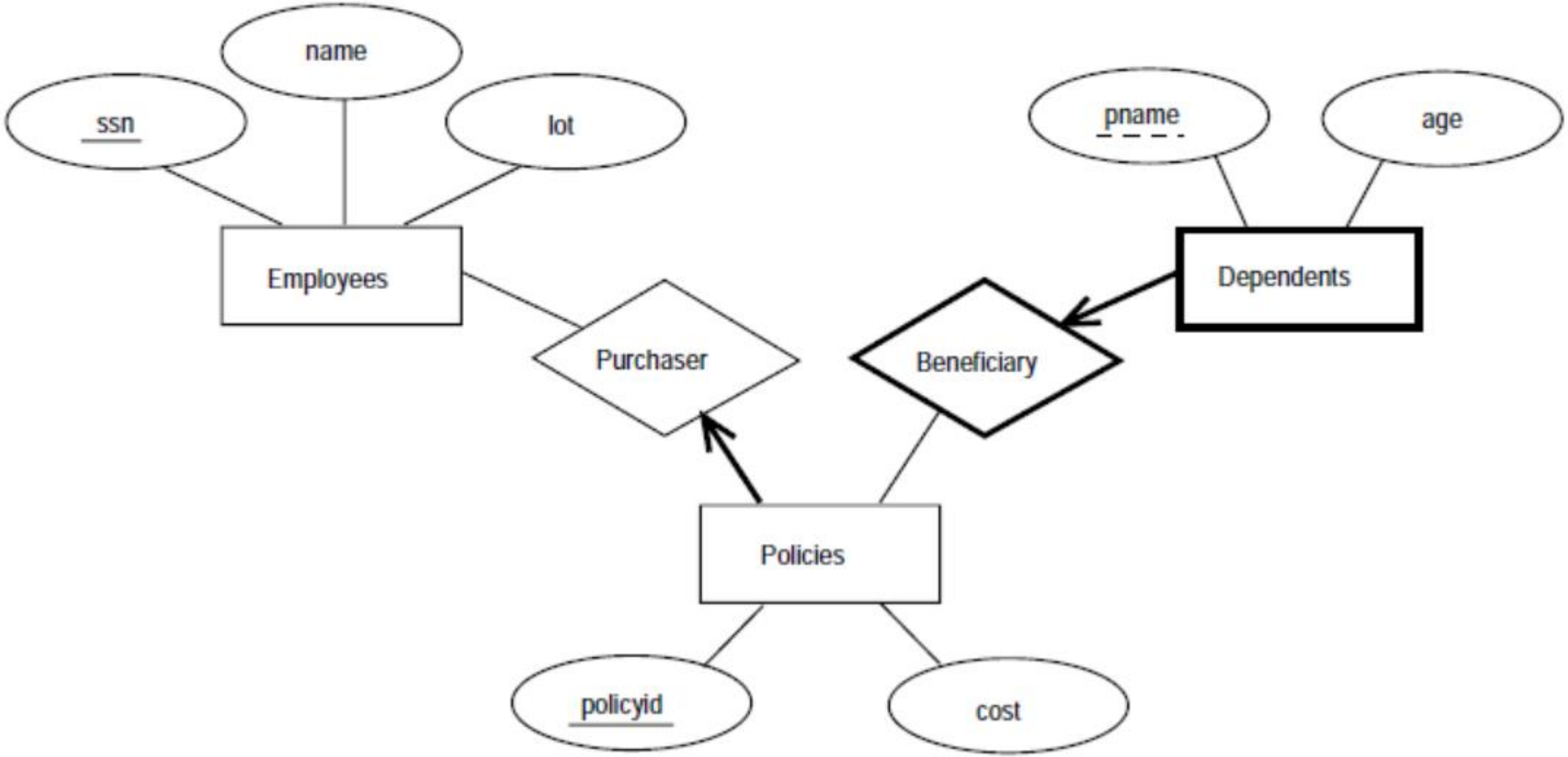
There is a special case in which this translation can be refined further by dropping the Sponsors relation. Consider the Sponsors relation. It has attributes *pid*, *did*, and *since*, and in general we need it (in addition to Monitors) for two reasons:

1. We have to record the descriptive attributes (in our example, *since*) of the Sponsors relationship.

2. Not every sponsorship has a monitor, and thus some $\langle pid, did \rangle$ pairs in the Sponsors relation may not appear in the Monitors relation.

However, if Sponsors has no descriptive attributes and has total participation in Monitors, every possible instance of the Sponsors relation can be obtained by looking at the $\langle pid, did \rangle$ columns of the Monitors relation. Thus, we need not store the Sponsors relation in this case.

ER to Relational



```
CREATE TABLE Policies ( policyid INTEGER,
                        cost     REAL,
                        ssn      CHAR(11) NOT NULL,
                        PRIMARY KEY (policyid),
                        FOREIGN KEY (ssn) REFERENCES Employees
                                ON DELETE CASCADE )
```

```
CREATE TABLE Dependents ( pname   CHAR(20),
                          age     INTEGER,
                          policyid INTEGER,
                          PRIMARY KEY (pname, policyid),
                          FOREIGN KEY (policyid) REFERENCES Policies
                                  ON DELETE CASCADE )
```

```
CREATE TABLE Dependents ( pname   CHAR(20),
                          ssn     CHAR(11),
                          age     INTEGER,
                          policyid INTEGER NOT NULL,
                          PRIMARY KEY (pname, policyid, ssn),
                          FOREIGN KEY (policyid, ssn) REFERENCES Policies
                                  ON DELETE CASCADE)
```