

Translating Entity Relationship to Relational Tables

EGCI321: LECTURE05 (WEEK 3)

E-R Diagram to Relational Schema

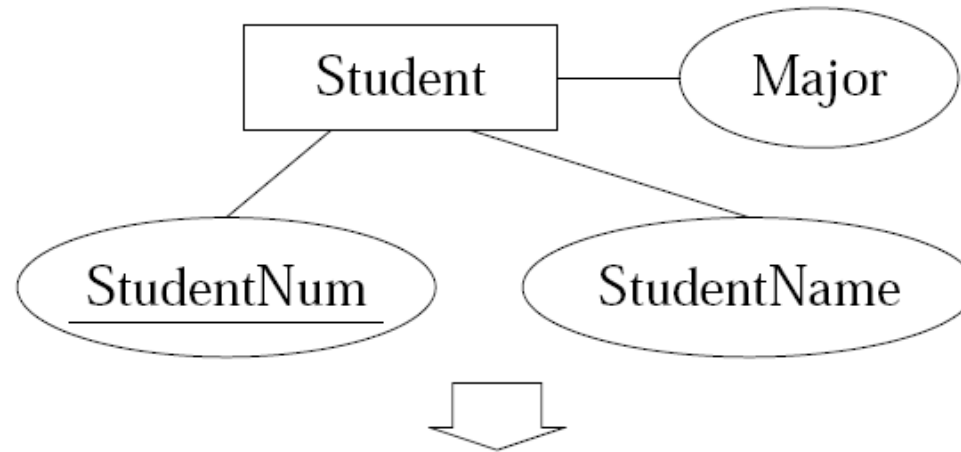
Main Idea:

- Each entity set maps to a new table
- Each attribute maps to a new table column
- Each relationship set maps to either new table columns or to a new table

Representing Strong Entity Sets

- Entity set E with attributes a_1, \dots, a_n translates to table E with attributes a_1, \dots, a_n
- Entity of type $E \leftrightarrow$ row in table E
- Primary key of entity set \rightarrow primary key of table

Example:



Student

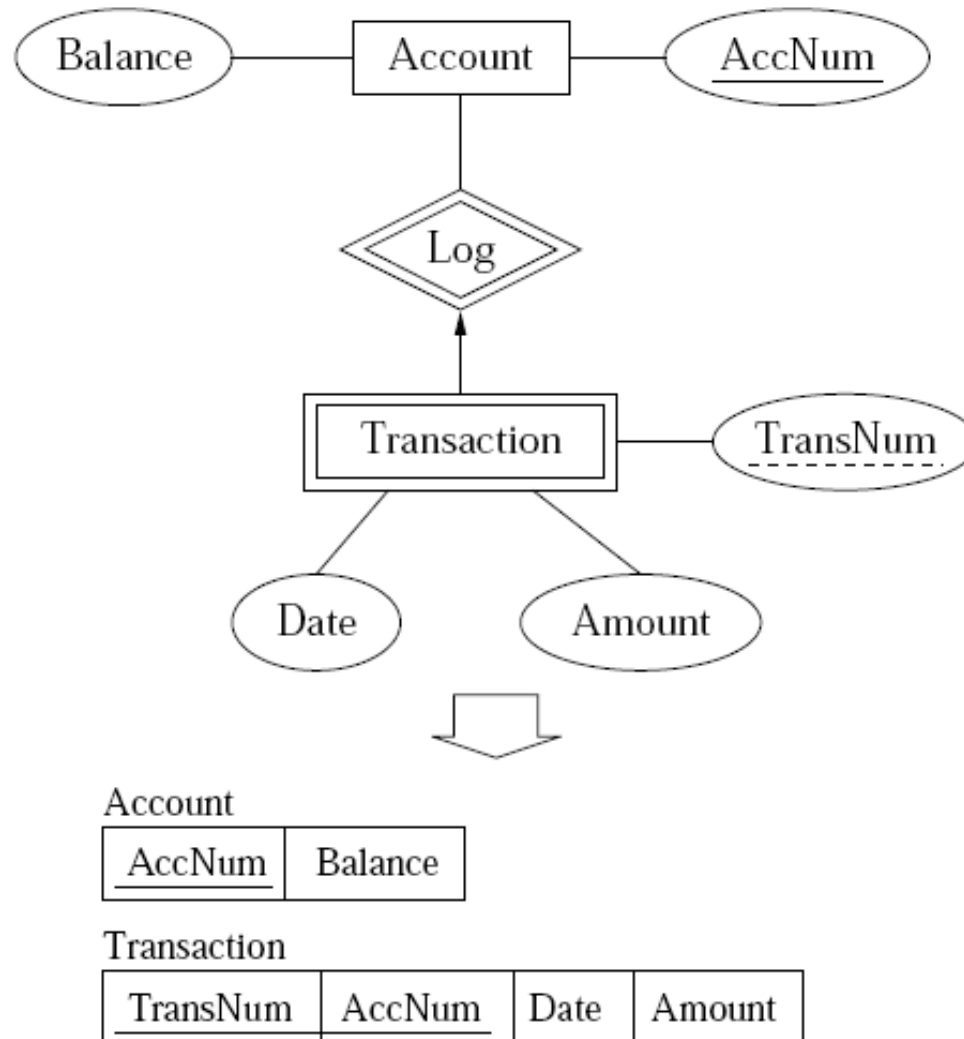
<u>StudentNum</u>	StudentName	Major
-------------------	-------------	-------

Representing Weak Entity Sets

- Weak entity set E translate to table E
- Columns of table E should include
 - Attributes of the weak entity set
 - Attributes of the identifying relationship set
 - Primary key attributes of entity set for dominating entities
- Primary key of weak entity \rightarrow set primary key of table

Representing Weak Entity Sets (cont.)

Example:



Representing Relationship Sets

- If the relationship set is an *identifying relationship* set for a *weak entity* set then no action needed
- If we can deduce the general cardinality constraint (1,1) for a component entity set *E* then add following columns to table *E*
 - Attributes of the relationship set
 - Primary key attributes of remaining component entity sets
- Otherwise: relationship set *R* → table *R*

Representing Relationship Sets (cont.)

Columns of table R should include:

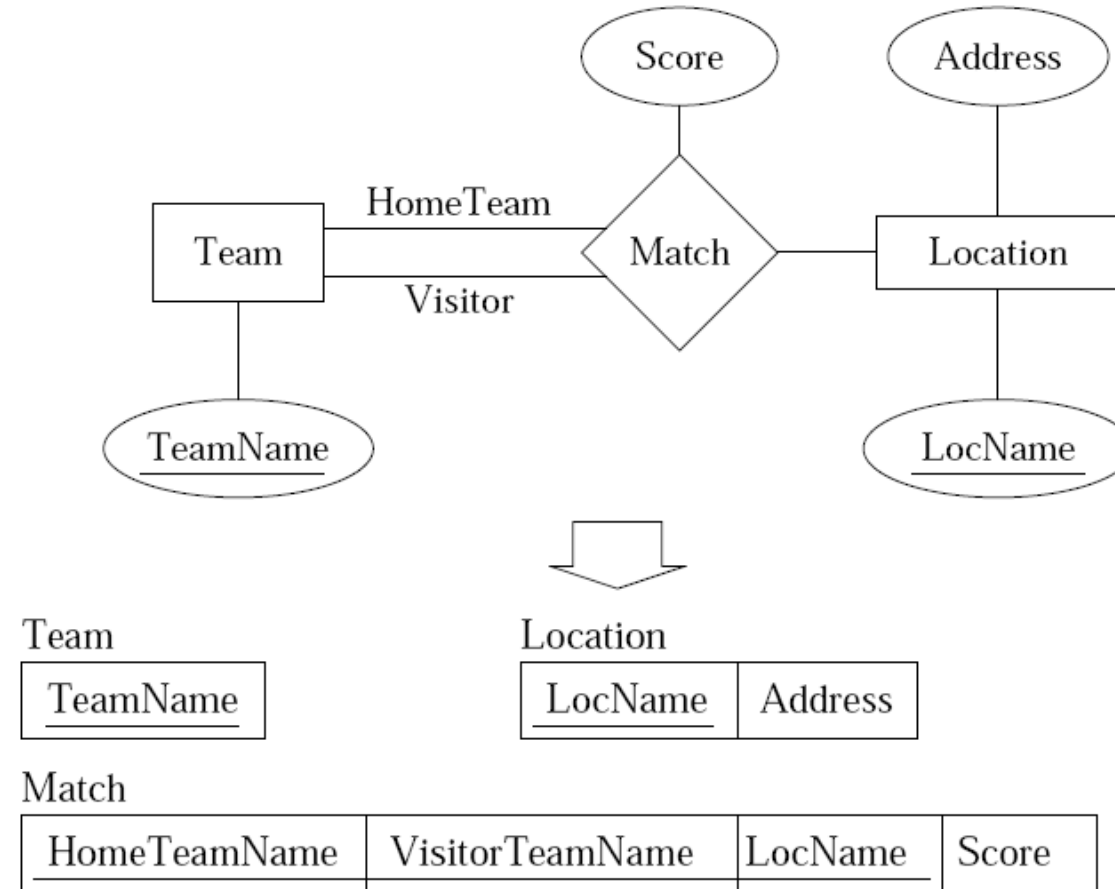
- Attributes of the relationship set
- Primary key attributes of each component entity set

Primary key of table R determined as follows

- If we can deduce the general cardinality constraint (0,1) for a component entity set E , then take the primary key attributes for E
- Otherwise, choose primary key attributes of each component entity

Representing Relationship Sets (cont.)

Example:



Note that the role name of a component entity set should be prepended to its primary key attributes, it supplied

Representing Aggregation

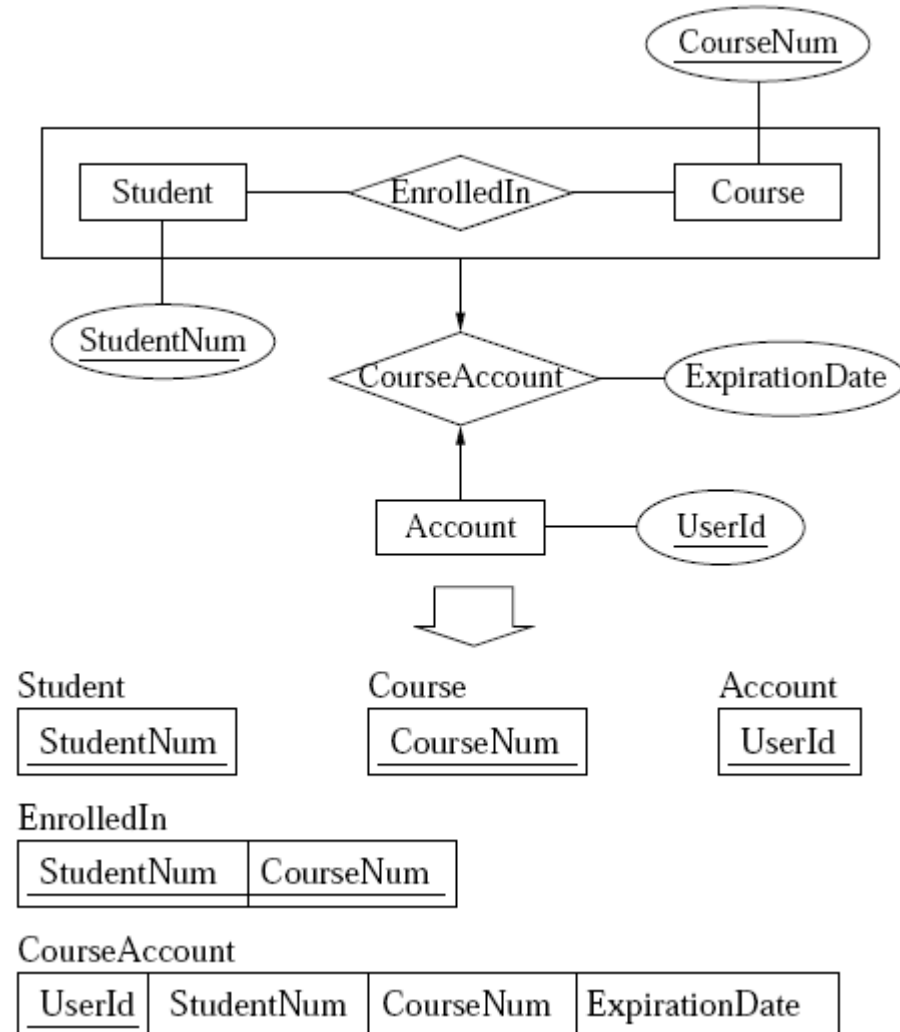
Tabular representation of aggregation of R

= tabular representation for relationship R

To represent relationship set involving aggregation of R , treat the aggregation like an entity set whose primary key is the **primary key** of the table for R

Representing Aggregation (cont.)

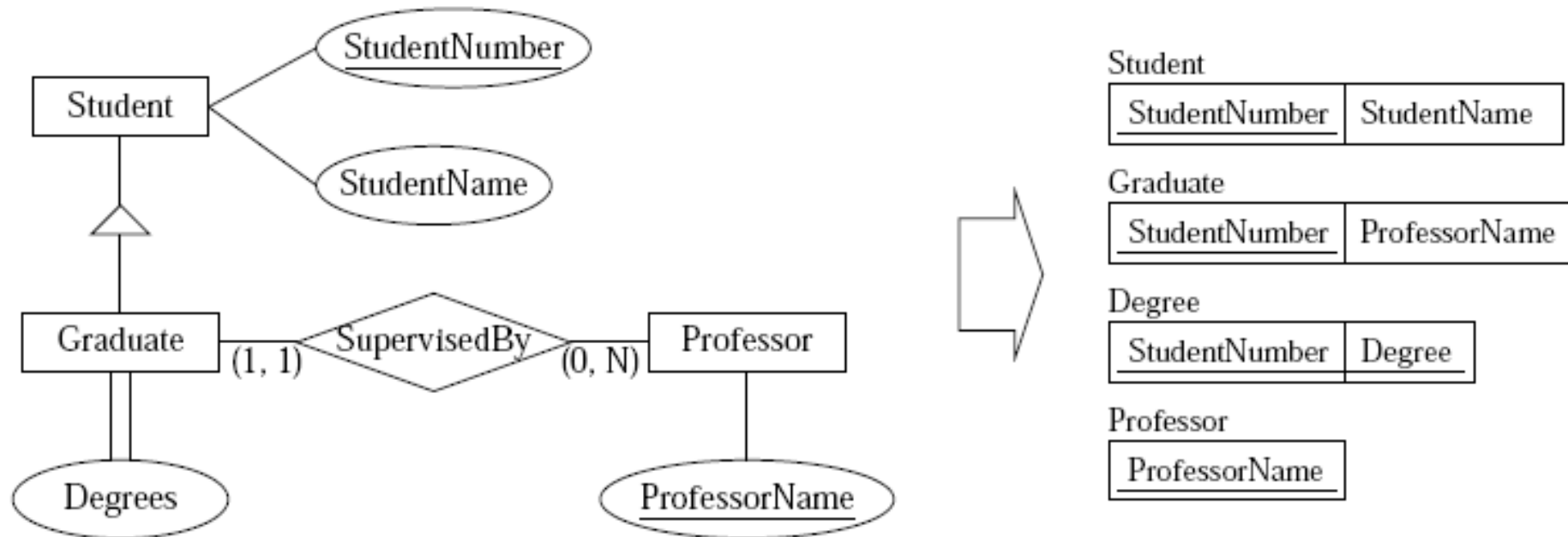
Example:



Representing Specialization

- Create table for higher-level entity set, and treat specialized entity subsets like weak entity sets (without discriminators)

Example:



Representing Generalization (Approach #1)

Create a table for each lower-level entity set only

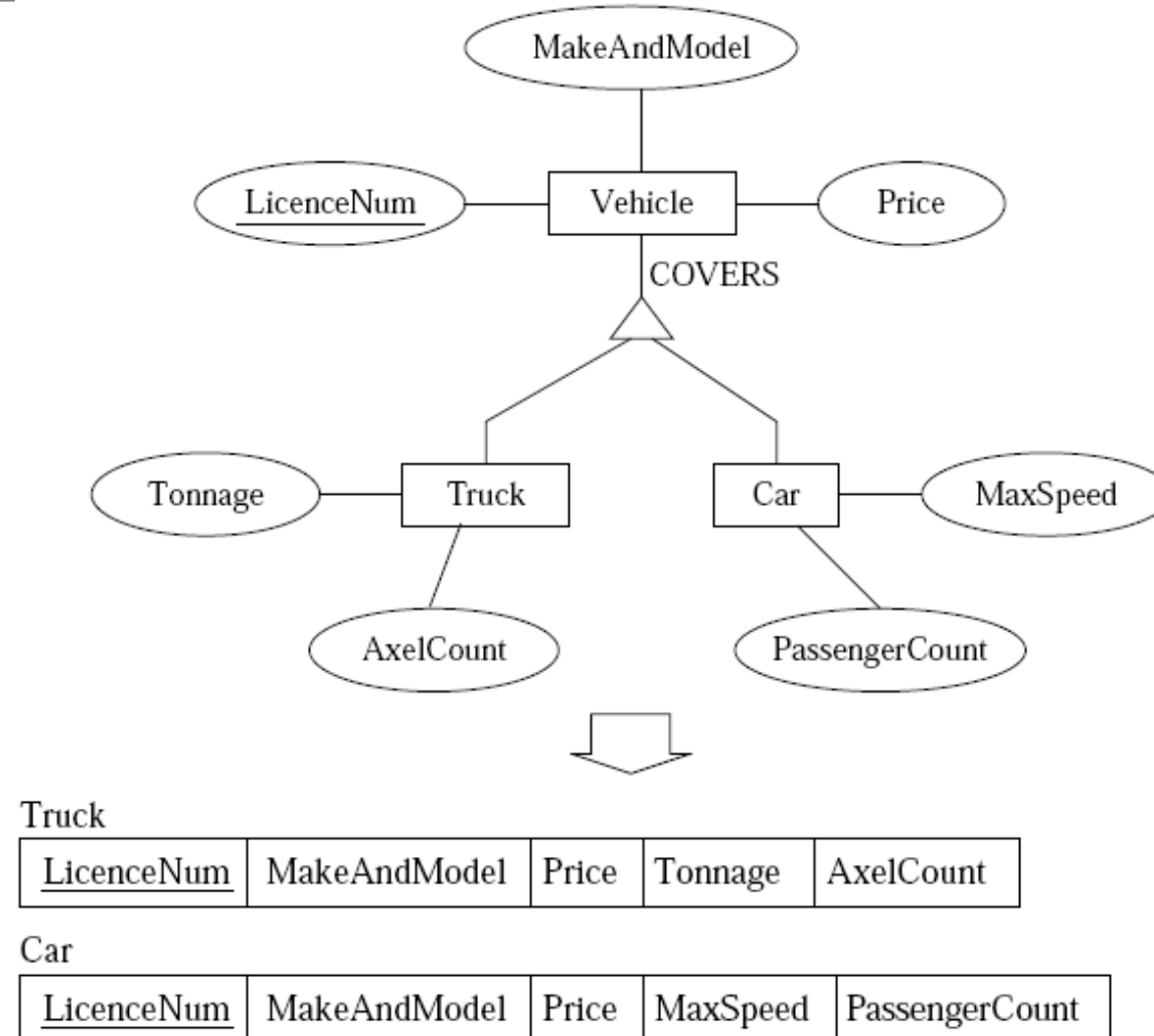
Columns of new tables should include:

- Attributes of lower level entity set
- Attributes of the superset

The higher level entity set can be defined as a view on the tables for the lower-level entity sets

Representing Generalization (Approach #1)

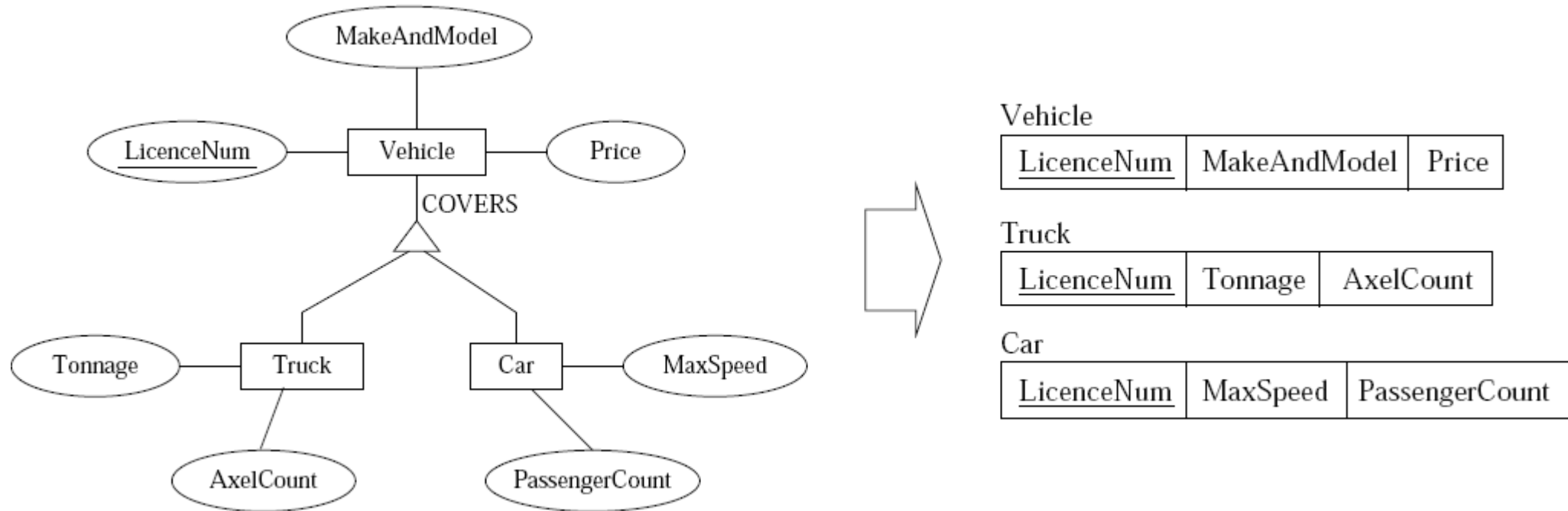
Example:



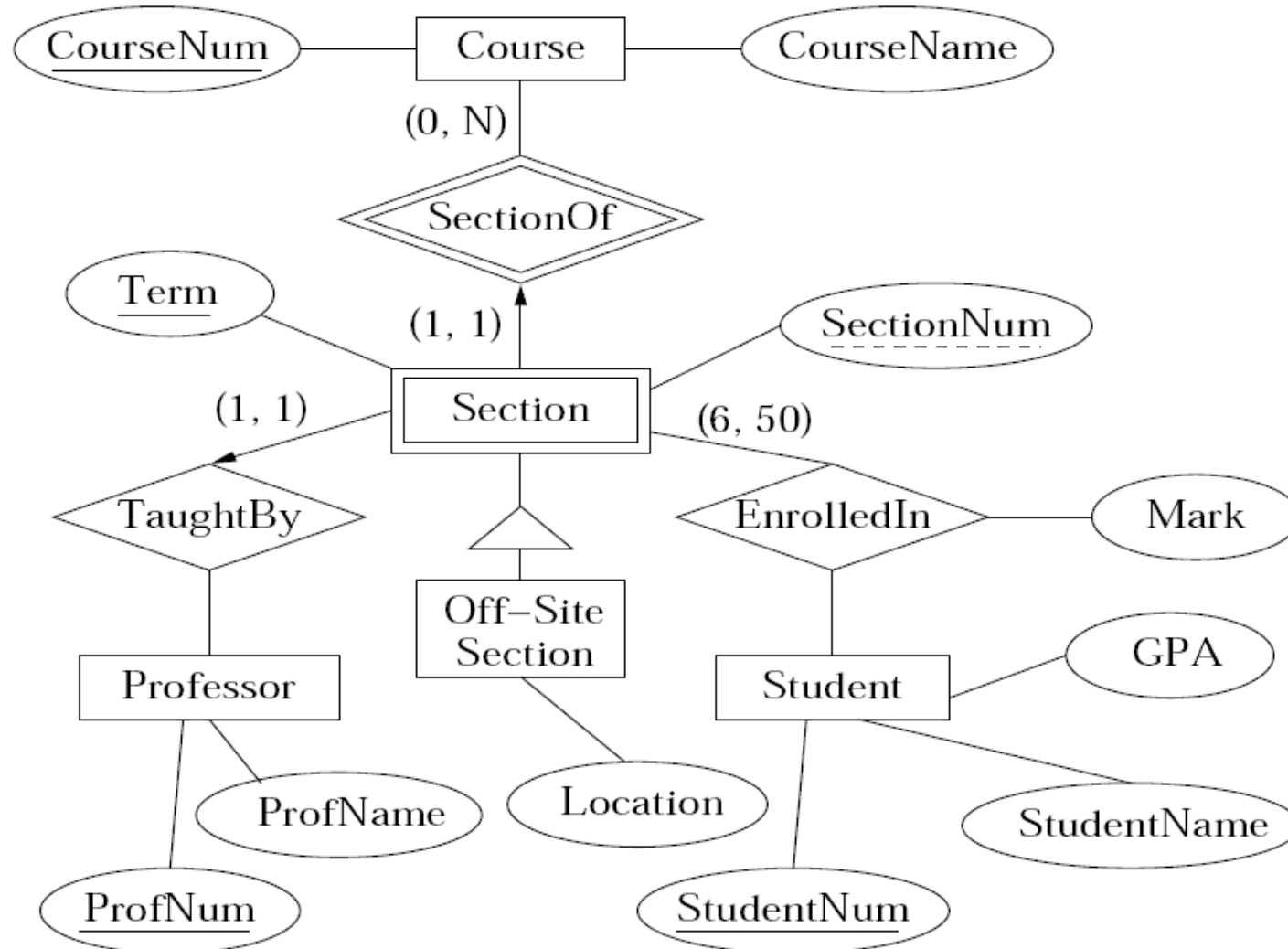
Representing Generalization (Approach #2)

Treat generalization the same as specialization

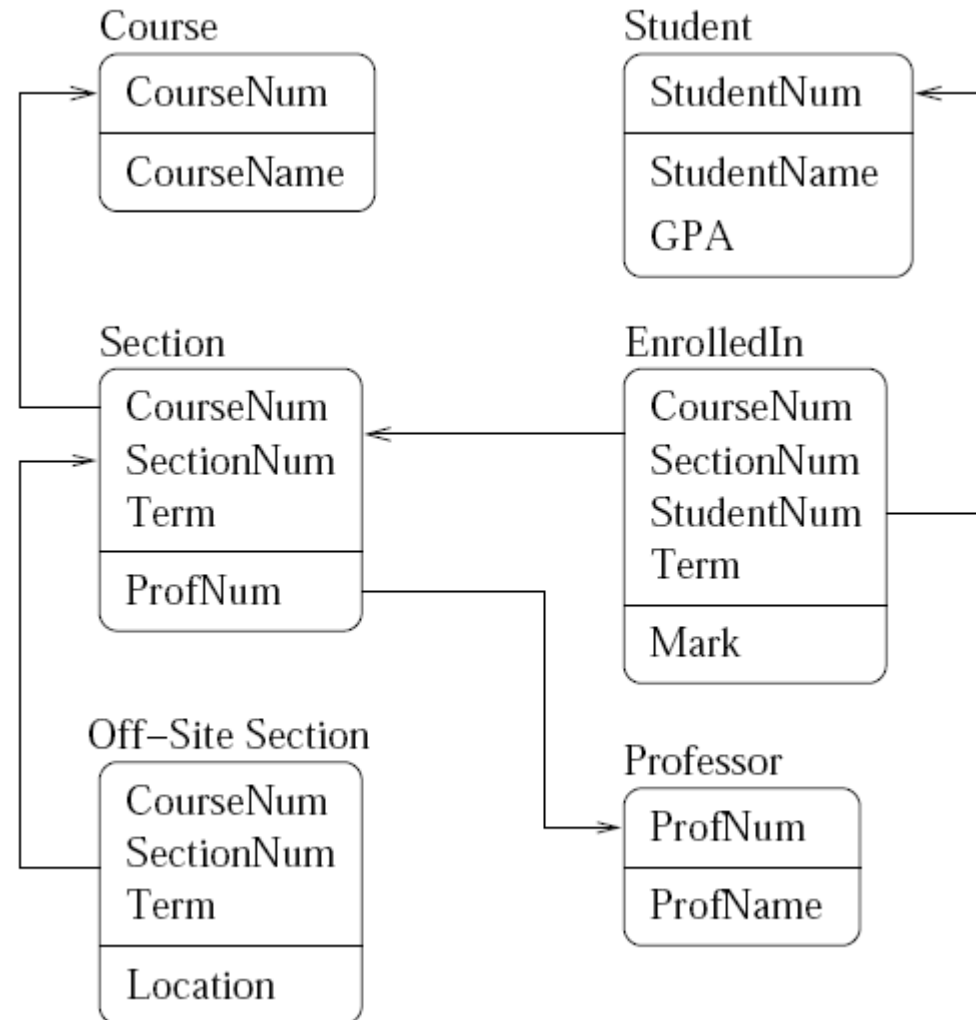
Example:



Example Translation: ER Diagram



Example Translation: Relational Diagram



Rational Model in SQL

SECTION II

Rational Model in SQL

- Creates the Students relation.

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

```
CREATE TABLE Students (sid NCHAR(20), name NCHAR(20), login NCHAR(10), age  
    INTEGER, gpa REAL)
```

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

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

Destroying and Altering Relations

- Destroys the relation Students. The schema information and the *tuples* are deleted.

```
DROP TABLE Students
```

- The schema of Students is altered by adding a new field; every tuple in the current instance is extended with a *null* value in the new field.

```
ALTER TABLE Students
```

```
ADD firstYear integer
```

Adding and Deleting Tuples

- Insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
VALUES (53688, 'Smith', 'smith@ee', 18, 3.2)
```

- Delete all tuples satisfying some condition (e.g., name = Smith):

```
SET SQL_SAFE_UPDATES = 0;
DELETE
FROM Students as S
WHERE S.name = 'Smith'
```

Review: 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!

Review: Primary Key Constraints

A set of fields is a key for a relation if :

1. No two distinct tuples can have same values in all key fields, and
2. This is not true for any subsets of the key.
 - Part 2 false? A *superkey*.
 - If there's >1 key 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 NCHAR(20), cid NCHAR(20), grade NCHAR(2), PRIMARY KEY (sid,cid) )
```

```
CREATE TABLE Enrolled (sid NCHAR(20), cid NCHAR(20), grade NCHAR(2), PRIMARY KEY (sid), UNIQUE (cid, grade) )
```

Used carelessly, an IC can prevent the storage of database instances that arise in practice!

Foreign Keys in SQL

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

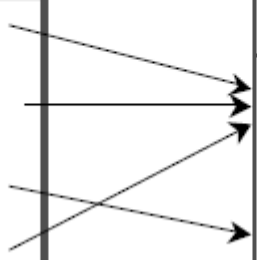
```
CREATE TABLE Enrolled (sid NCHAR(20), cid NCHAR(20), grade NCHAR(2),  
PRIMARY KEY (sid,cid), FOREIGN KEY (sid) REFERENCES Students(sid) )
```

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



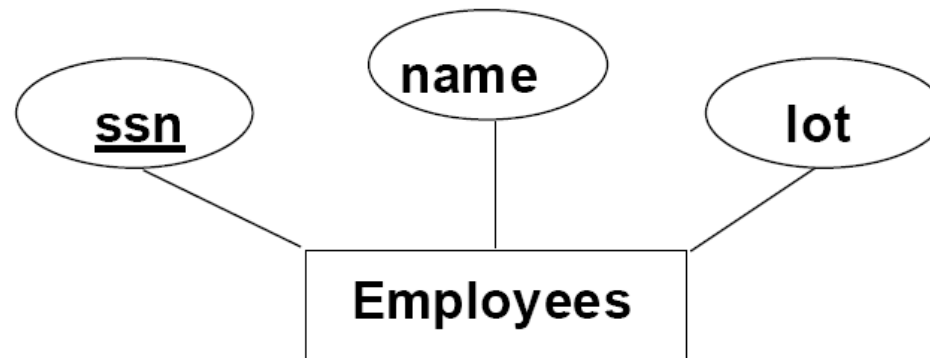
Referential Integrity in SQL

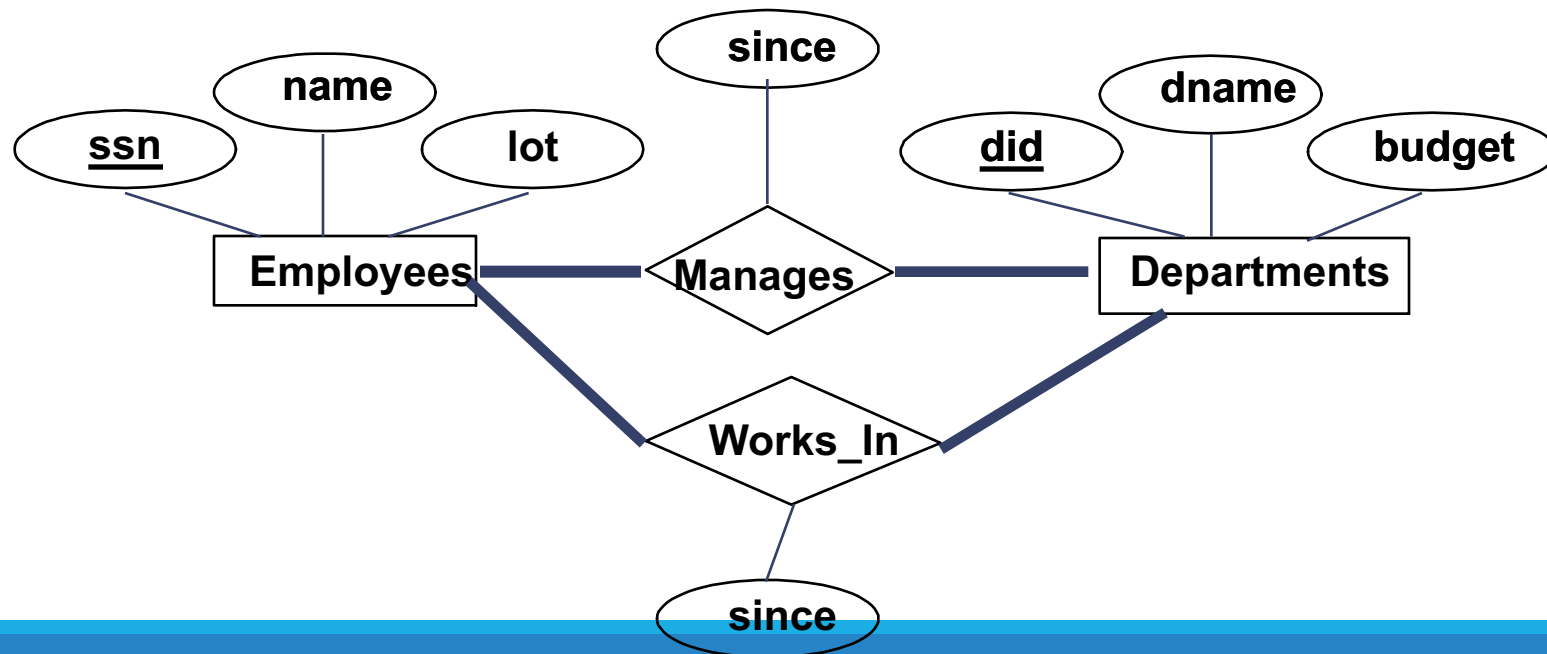
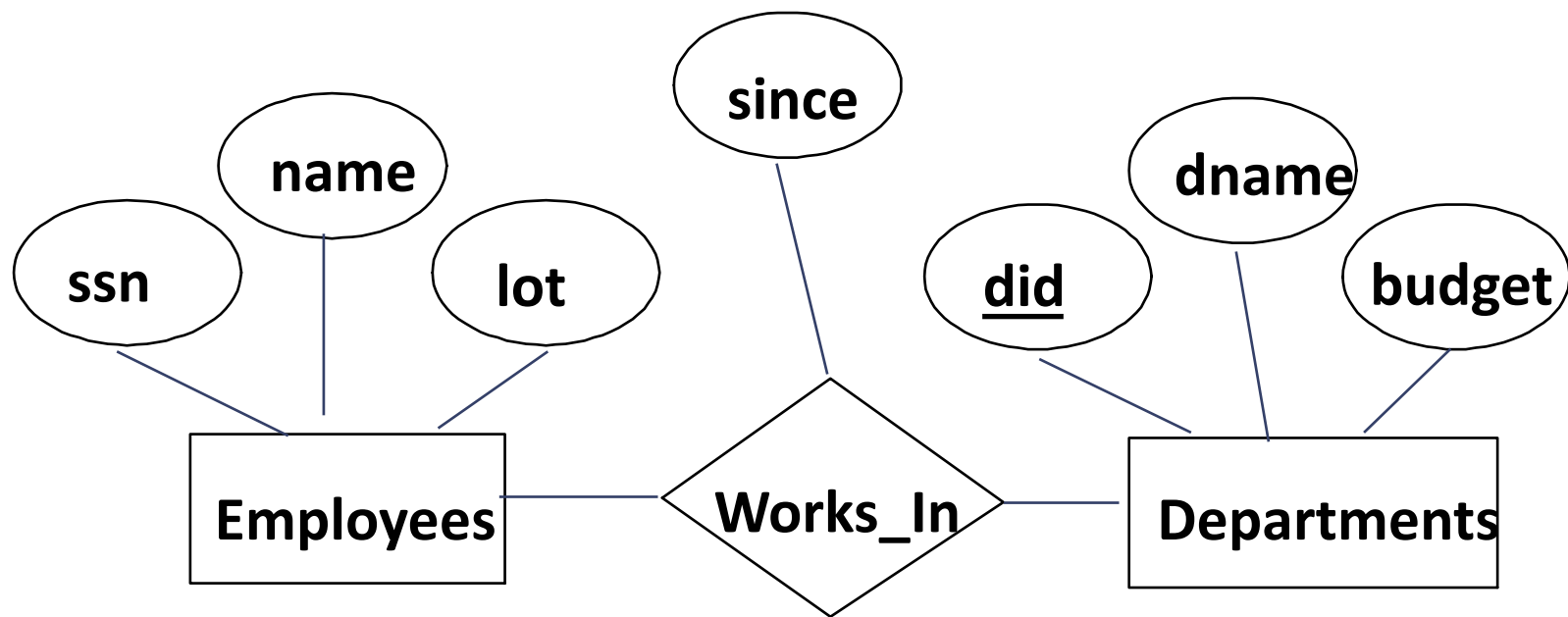
```
CREATE TABLE Enrolled  
(sid NCHAR(20),  
cid NCHAR(20),  
grade NCHAR(2),  
PRIMARY KEY (sid,cid),  
FOREIGN KEY (sid)  
REFERENCES Students(sid))
```

ER to Relational

Entity sets to tables:

```
CREATE TABLE Employees (ssn CHAR(11), name NCHAR(20), lot INTEGER,  
PRIMARY KEY (ssn))
```





Relationship Sets to Tables

In translating a relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys).
 - ▶ This set of attributes forms a *superkey* for the relation.

All descriptive attributes.

```
CREATE TABLE Works_In( ssn NCHAR(11), did NCHAR(11), since DATE,  
PRIMARY KEY (ssn, did), FOREIGN KEY (ssn) REFERENCES Employees(ssn),  
FOREIGN KEY (did) REFERENCES Departments(did))
```

Translating ER Diagrams with Key Constraints

Map relationship to a table:

- Note that did is the key now!
- Separate tables for Employees and Departments.

Since each department has a unique manager, we could instead combine Manages and Departments.

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

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

Reference

1. Ramakrishnan R, Gehrke J., Database management systems, 3rd ed., New York (NY): McGraw-Hill, 2003.