Module 6 - Mapping ER diagrams to tables

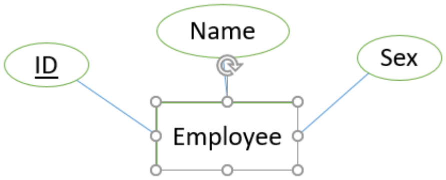
The ER model is convenient for representing an initial, high-level database design. Now we will see how we can translate an ER diagram into a collection of tables, that is, a relational database schema.

Entities to tables

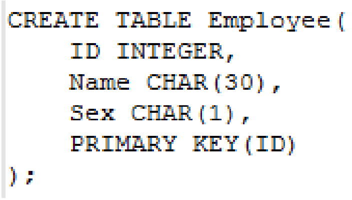
An entity is a real-world object with some attributes. We can map entities to tables by,

- Create table for each entity.
- Each attribute of the entity set becomes an attribute of the table, with their respective data types.
- Declare primary key.

For example, consider the “Employee” entity as shown below –



The following SQL statement captures the above information,

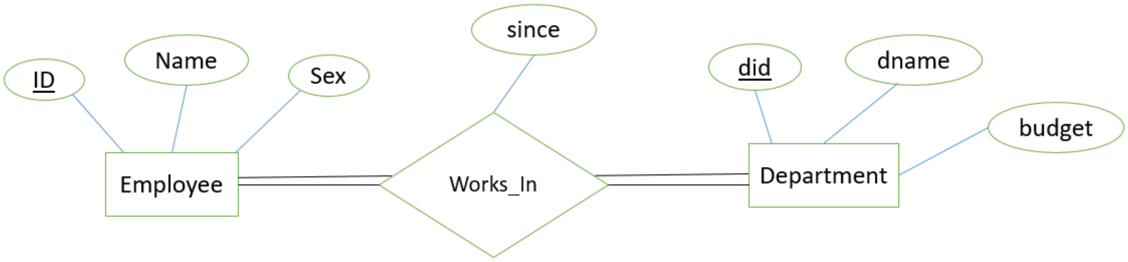


```
CREATE TABLE Employee(
    ID INTEGER,
    Name CHAR(30),
    Sex CHAR(1),
    PRIMARY KEY(ID)
);
```

Relationships to tables

- Create table for a relationship.
- Add the primary keys of all participating entities as fields of table with their respective data types.
- If relationship has any attribute, add each attribute as field of table.
- Declare a primary key composing all the primary keys of participating entities.
- Declare as foreign key fields the primary key attributes of each participating entity.

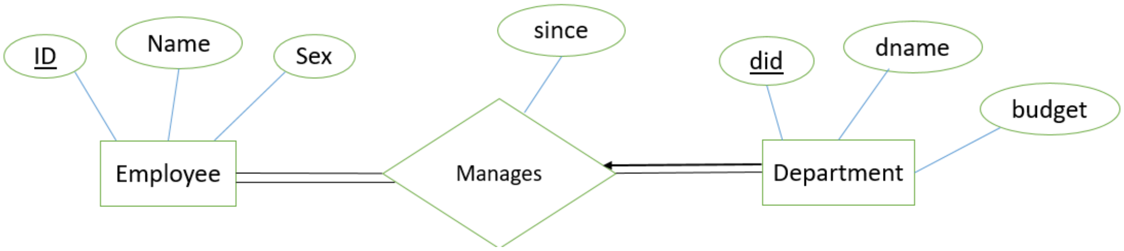
For example, consider the “Works_In” relationship as shown below -



The following SQL statement captures the above information,

```
CREATE TABLE Works_In(
    ID INTEGER,
    did INTEGER,
    since DATE,
    PRIMARY KEY (ID, did),
    FOREIGN KEY (ID) REFERENCES Employees(ID),
    FOREIGN KEY (did) REFERENCES Departments(did)
);
```

Consider the relationship “Manages” as shown below –



The table corresponding to Manages has the attributes *ID, did, since*. However, because each department has at most one manager, no two records can have the same *did* value but differ on the ID value. Hence *did* is itself a key for Manages and not the set *ID, did*.

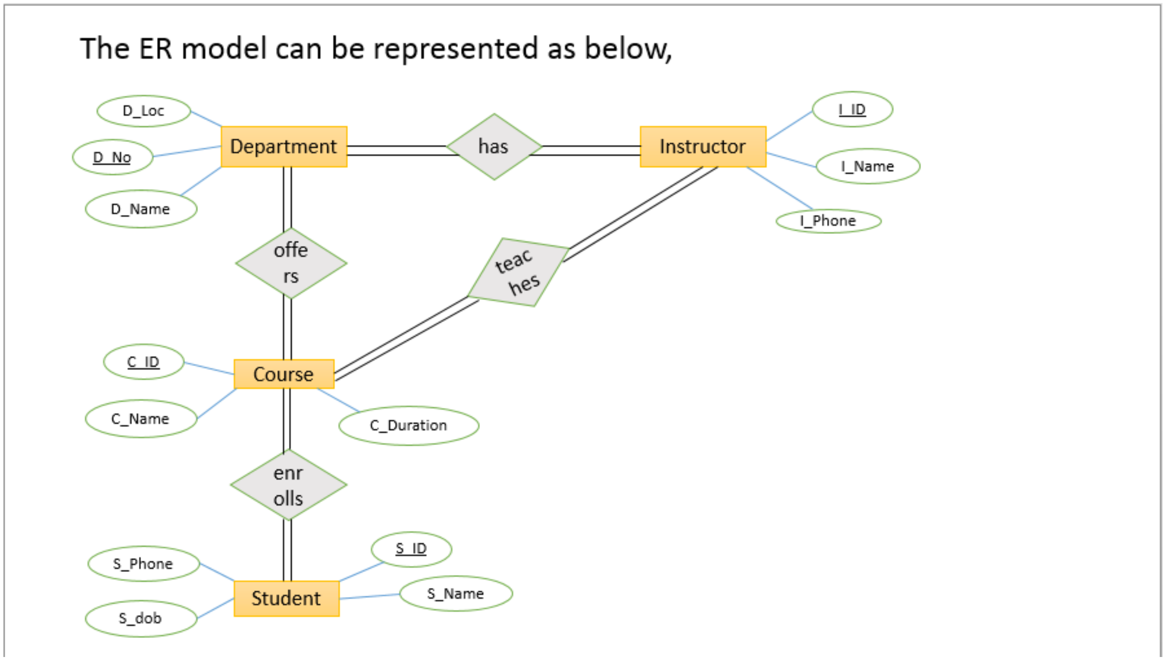
The Manages relation can be defined using the following SQL statement,

```
CREATE TABLE Manages(
    ID INTEGER,
    did INTEGER,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ID) REFERENCES Employees(ID),
    FOREIGN KEY (did) REFERENCES Departments(did)
);
```

Example

Consider the university ER model discussed in slide 13 of Day 2. Let us translate this ER diagram into a relational model by writing the SQL statements.

Let us import the ER diagram for convenience.



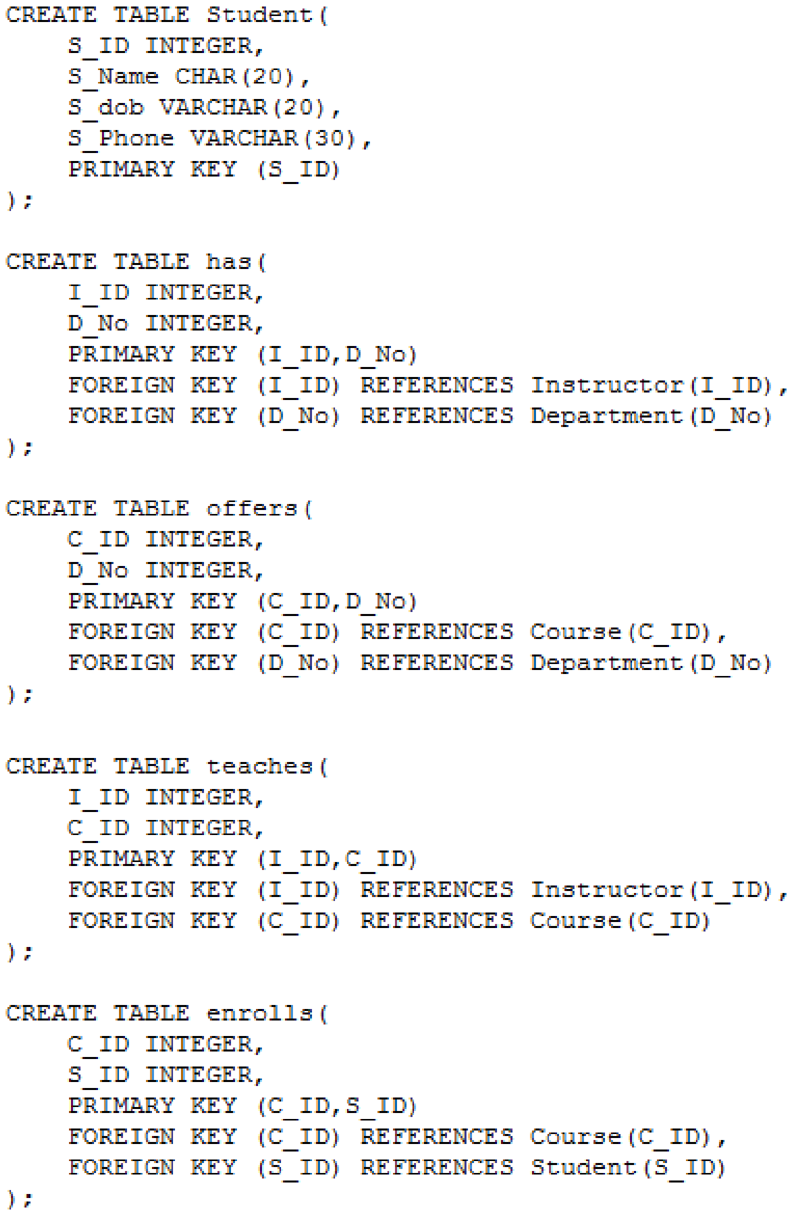
The ER model can be represented as below,

The following SQL statements captures the required information,

```
CREATE TABLE Department(
    D_No INTEGER,
    D_Name CHAR(20),
    D_Loc VARCHAR(30),
    PRIMARY KEY (D_No)
);

CREATE TABLE Instructor(
    I_ID INTEGER,
    I_Name CHAR(20),
    I_Phone VARCHAR(30),
    PRIMARY KEY (I_ID)
);

CREATE TABLE Course(
    C_ID INTEGER,
    C_Name CHAR(20),
    C_Duration VARCHAR(30),
    PRIMARY KEY (C_ID)
);
```



```
CREATE TABLE Student(
    S_ID INTEGER,
    S_Name CHAR(20),
    S_dob VARCHAR(20),
    S_Phone VARCHAR(30),
    PRIMARY KEY (S_ID)
);

CREATE TABLE has(
    I_ID INTEGER,
    D_No INTEGER,
    PRIMARY KEY (I_ID,D_No)
    FOREIGN KEY (I_ID) REFERENCES Instructor(I_ID),
    FOREIGN KEY (D_No) REFERENCES Department(D_No)
);

CREATE TABLE offers(
    C_ID INTEGER,
    D_No INTEGER,
    PRIMARY KEY (C_ID,D_No)
    FOREIGN KEY (C_ID) REFERENCES Course(C_ID),
    FOREIGN KEY (D_No) REFERENCES Department(D_No)
);

CREATE TABLE teaches(
    I_ID INTEGER,
    C_ID INTEGER,
    PRIMARY KEY (I_ID,C_ID)
    FOREIGN KEY (I_ID) REFERENCES Instructor(I_ID),
    FOREIGN KEY (C_ID) REFERENCES Course(C_ID)
);

CREATE TABLE enrolls(
    C_ID INTEGER,
    S_ID INTEGER,
    PRIMARY KEY (C_ID,S_ID)
    FOREIGN KEY (C_ID) REFERENCES Course(C_ID),
    FOREIGN KEY (S_ID) REFERENCES Student(S_ID)
);
```