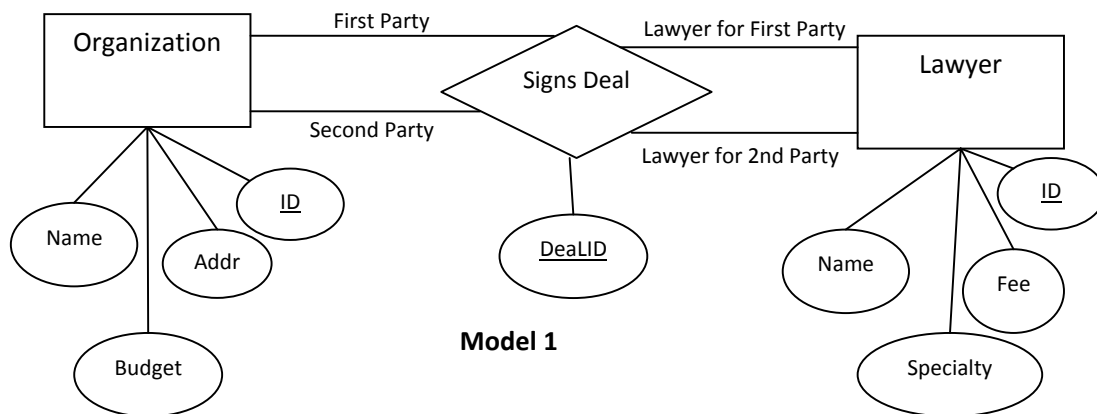


Homework1

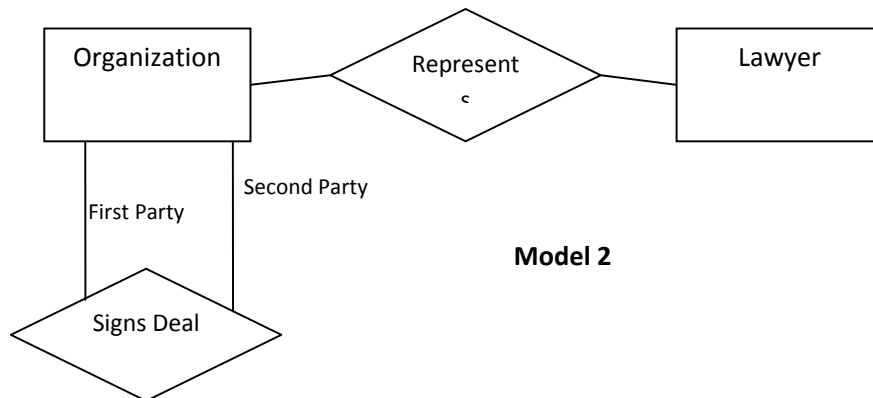
4.14

- Drop **Table WHENOFFERED**.
- Drop the foreign key constraint of WHENOFFERED in Table CLASS.
- Add **CREATE DOMAIN SEMESTER CHAR(6)**
CHECK (VALUE IN ('SPRING', 'FALL', ' BOTH'))
- Replace **Semester Char(6)** with **SemesterofClass SEMESTER** in Table **CLASS**, add **SemesterofCourse SEMESTER** in Table **COURSE**.
- Add constraint in Table CLASS:
CHECK (Semester IN ("Spring","Fall"))
- Add constraint in Table CLASS:
CONSTRAINT SemesterOffered
CHECK (0 < (SELECT COUNT(*) FROM COURSE A
WHERE
A.CrsCode = CrsCode AND
(A. SemesterofClass = 'BOTH' OR A.SemesterofClass = SemesterofCourse)))

4.15



Information loss can occur if E-R model #1 is transformed into model #2, which has two binary relations instead of one 4th degree relation. In E-R model #2, it is impossible to tell which lawyer was present during a specific deal (because each organization can have multiple lawyers). That information is encoded in model #1.



Model 2

If we assume there is a one to one relationship between organizations and lawyers, then no information loss occurs when moving from model #1 to model #2. That's because for every deal, each organization uses the same lawyer; that information can be extracted from model #2.

Schema for the relational representation of model #1:

ORGANIZATION(OrgID, OrgName, Address, Budget)

LAWYER(LawyerID, Name, Specialty, Fee)

DEAL(DealID, OrgID1, OrgID2, LawyerID1, LawyerID2)

In the relation **DEAL**, *OrgID1*, *OrgID2*, *LawyerID1*, and *LawyerID2* are all foreign keys. *DealID* is the primary key because the tuple $\langle \text{OrgID1}, \text{OrgID2}, \text{LawyerID1}, \text{LawyerID2} \rangle$ is not necessarily unique.

Homework2

- 5.20 TRANSCRIPT(StudId, CrsCode, Semester, Grade)
TEACHING(ProfId, CrsCode, Semester)
PROFESSOR(Id, ProfName, Dept)

$$\pi_{StudId} \left[\frac{\pi_{StudId, ProfId}(TRANSCRIPT \bowtie TEACHING)}{\pi_{Id}(\sigma_{Dept="MUS"}PROFESSOR)} \right]$$

```
SELECT DISTINCT S.StudId
FROM Transcript S
WHERE
    NOT EXISTS ( -- All Professors in MUS dept
        (SELECT ProfId
         FROM Professor
         WHERE Dept="MUS")
    EXCEPT -- All Professors the student has taken a class with
        (SELECT R.ProfId
         FROM Transcript T, Teaching R
         WHERE S.StudId = T.StudId
              AND T.CrsCode = R.CrsCode
              AND T.Semester = R.Semester) )
```

- 5.21 CREATE VIEW MUS_Students AS (...)
"..." is the SQL answer from 5.20

```
CREATE VIEW NumCourses AS (
    SELECT T.StudId, COUNT(*) AS NumCrs
    FROM Transcript T
    GROUP BY T.StudId )
```

```
SELECT S.StudId, T.NumCrs
FROM MUS_Students S, NumCourses T
WHERE S.StudId = T.StudId AND T.NumCrs > 10
```

5.22 BROKER(Id, Name)
 ACCOUNT(Acct#, BrokerId, Gain)

$$\pi_{Name} Broker - \left(\pi_{Name} \left[Broker_{Id = BrokerId} (\sigma_{Gain \leq 0} Account) \right] \right)$$

```

SELECT B.Name
FROM Broker B
WHERE
  NOT EXISTS (
    SELECT *
    FROM Account A
    WHERE B.Id = A.BrokerId
    AND A.Gain <= 0 )

```

5.23 CREATE VIEW NoGain AS (
 SELECT *
 FROM Account
 WHERE Gain <= 0)

```

CREATE VIEW AccountCnt AS (
  SELECT A.BrokerId, COUNT(*) AS Cnt
  FROM Account A
  GROUP BY A.BrokerId )

```

```

CREATE VIEW NoGainCnt AS (
  SELECT A.BrokerId, COUNT(*) AS Cnt
  FROM NoGain A
  GROUP BY A.BrokerId )

```

```

DELETE FROM Broker
WHERE Id IN (SELECT A.BrokerId
  FROM AccountCnt A, NoGainCnt B
  WHERE A.BrokerId = B.BrokerId
  AND (B.Cnt / A.Cnt) >= 0.4 )

```

5.24

(a)

```
SELECT C.ID FROM CUSTOMER C WHERE  
FOR ALL (  
(SELECT P.FEATURE FROM PREFERENCE P WHERE P.CUSTID = C.ID)  
P.FEATURE IN  
(SELECT A.FEATURE FROM AMNEITY A, HOUSE H  
WHERE A.ADDRESS=H.ADDRESS AND H.ANGETID='007'))
```

(b)

```
CREATE VIEW InterestedCustomers as (a)  
SELECT P.FEATURE, COUNT (P.CUSTID )  
FROM PREFERENCE P  
WHERE P.CUSTID IN (SELECT ID FROM InterestedCustomers)  
HAVING COUNT(P.CUSTID)>3  
GROUP BY P.FEATURE
```

```
7.6 CREATE TRIGGER DropCourseCheck  
      AFTER DELETE ON Transcript  
      FOR EACH ROW  
      ...  
  
CREATE TRIGGER ChangeMajorCheck  
      AFTER UPDATE OF Major ON Student  
      FOR EACH ROW  
      ...  
  
CREATE TRIGGER GpaDropCheck  
      AFTER INSERT, UPDATE OF Grade ON Transcript  
      REFERENCING NEW AS N  
      FOR EACH ROW  
      WHEN (grade_avg(N.StudId) <= threshold)  
      ...
```