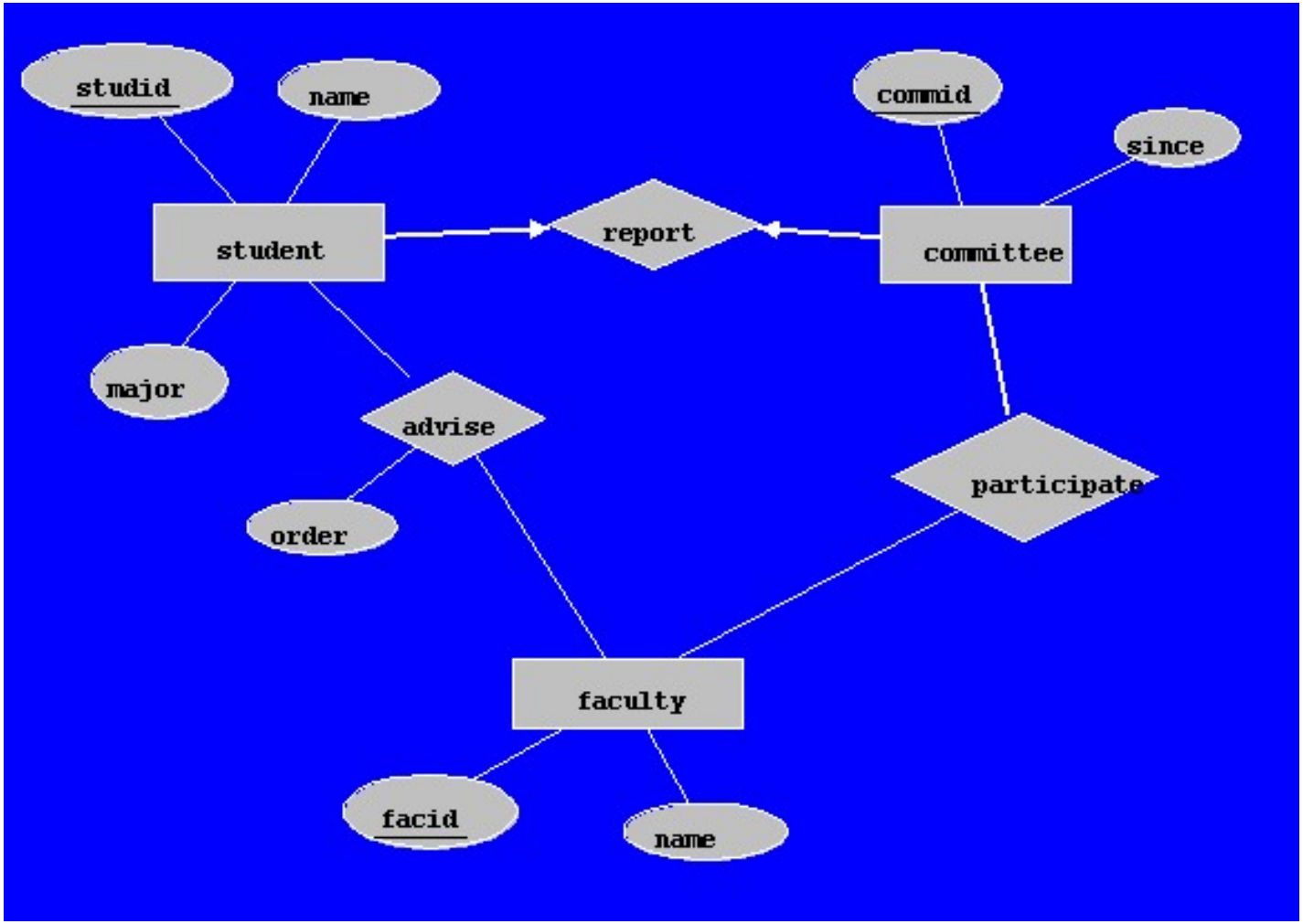
Database Design

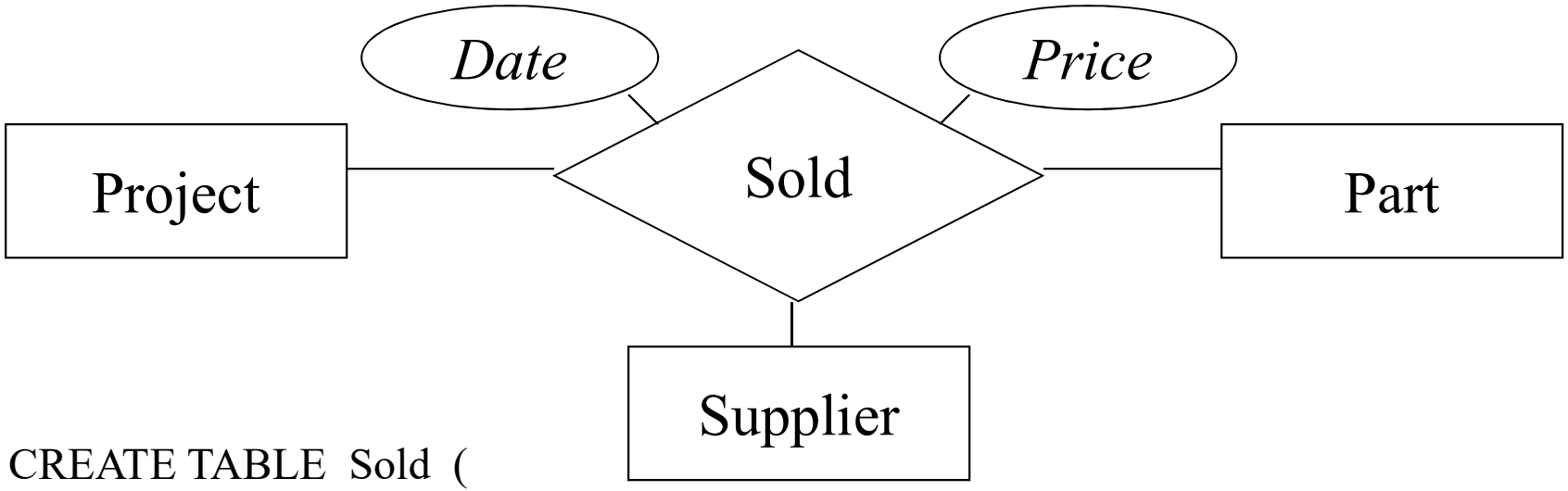
- Goal: specification of database schema
- Methodology:
  - Use *E-R model* to get a high-level graphical view of essential components of enterprise and how they are related
  - Convert E-R diagram to DDL
- *E-R Model*: enterprise is viewed as a set of
  - *Entities*
  - *Relationships* among entities

A Sample ER Diagram



Representation in SQL

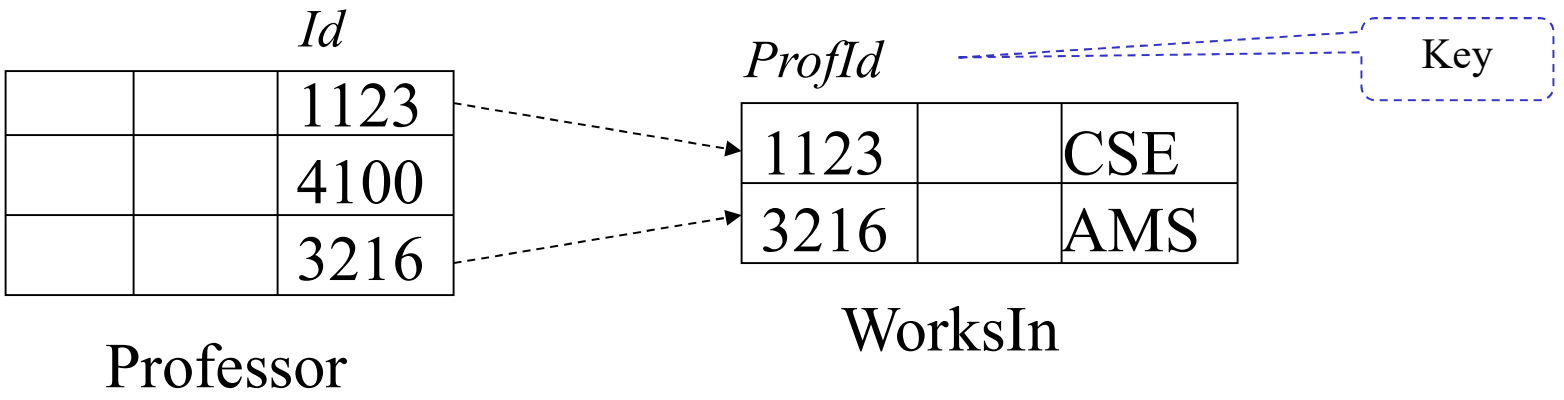
- Each role of relationship type produces a foreign key in corresponding relation
  - Foreign key references table corresponding to entity type from which role values are drawn



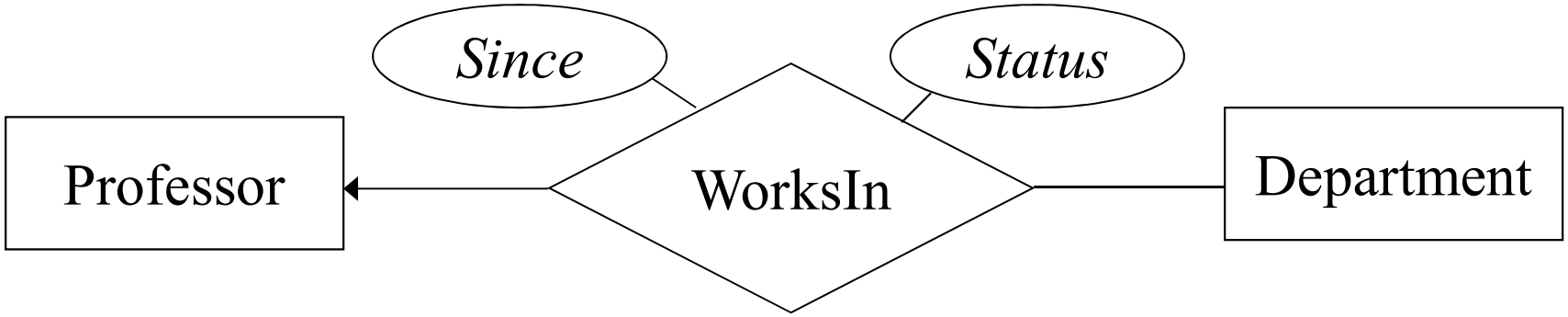

```
CREATE TABLE  Sold  (
   Price  INTEGER,              -- attribute
   Date  DATE,                  -- attribute
   ProjId  INTEGER,             -- role
   SupplierId  INTEGER,         -- role
   PartNumber  INTEGER,         -- role
   PRIMARY KEY (ProjId, SupplierId, PartNumber, Date),
   FOREIGN KEY (ProjId) REFERENCES Project,
   FOREIGN KEY (SupplierId) REFERENCES Supplier (Id),
   FOREIGN KEY (PartNumber) REFERENCES Part (Number)  )
```

Key Constraints

- Each professor works in AT MOST ONE department



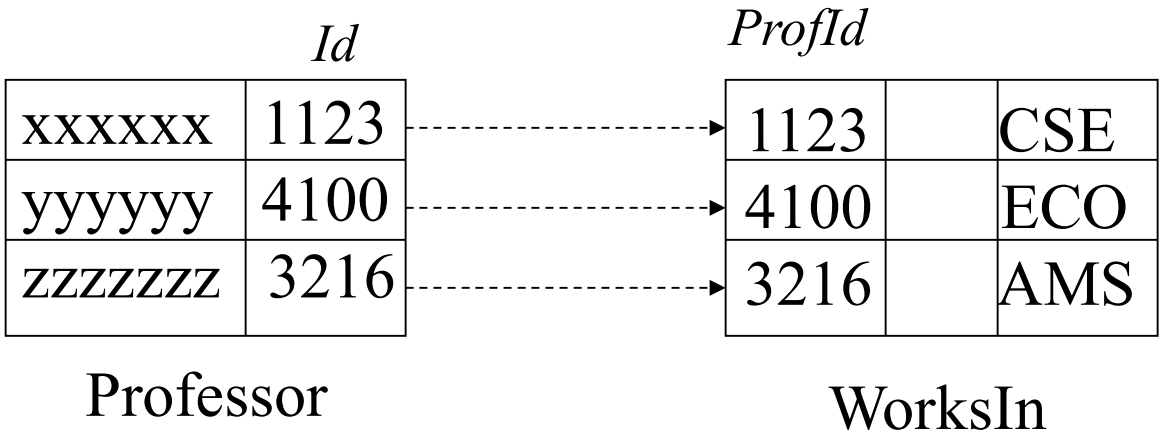
Key Constraints



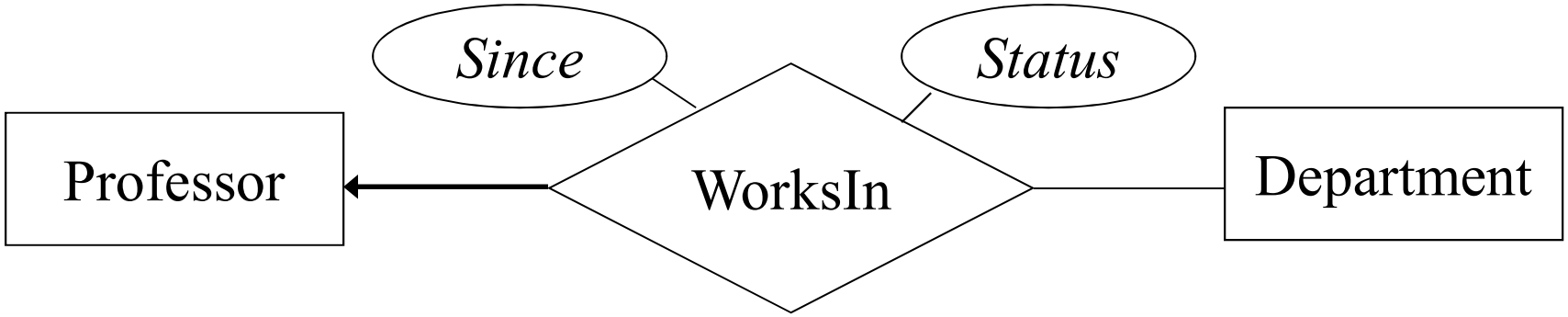
CREATE TABLE WorksIn (
ProfId INTEGER, -- *role (key of* Professor*)*
Since DATE, -- *attribute*
Status CHAR (10), -- *attribute*
DeptId CHAR (4), -- *role (key of* Department*)*
PRIMARY KEY (*ProfId*), -- *since a professor works in at most one department*
FOREIGN KEY (*ProfId*) REFERENCES Professor (*Id*),
FOREIGN KEY (*DeptId*) REFERENCES Department)

Key and Total Participation Constraints

- Each professor works in EXACTLY ONE department



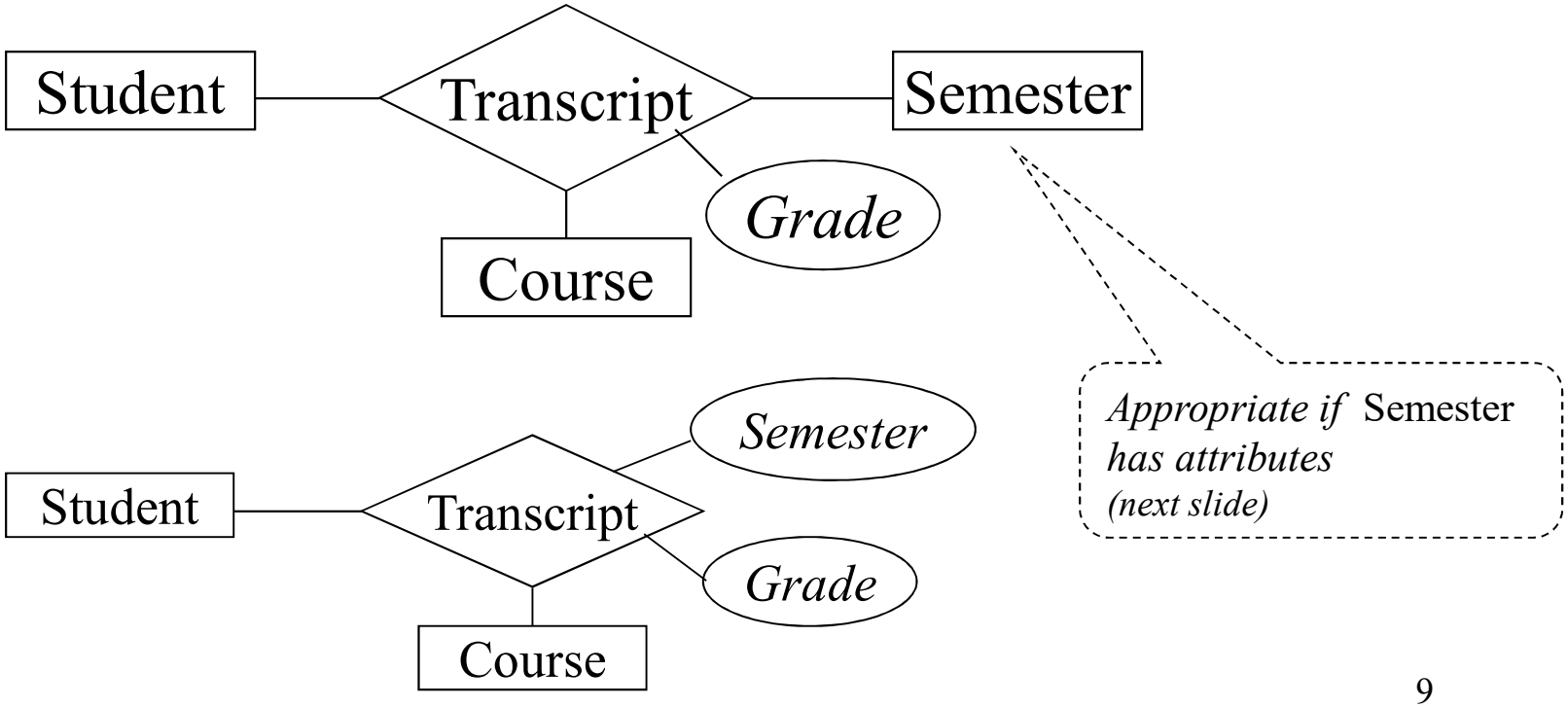
Key + Total Participation Constraints



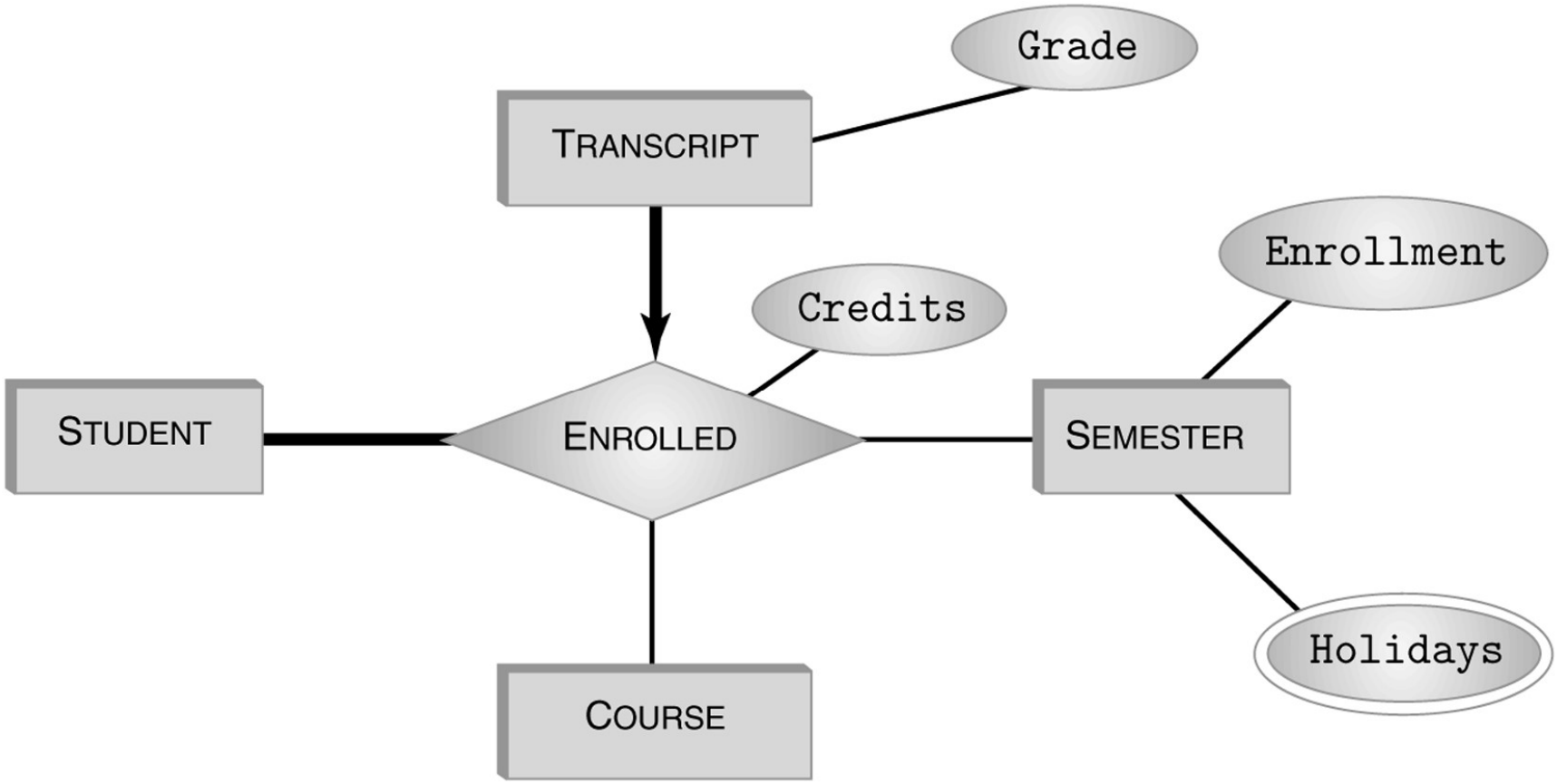
CREATE TABLE WorksIn (
ProfId INTEGER, -- *role (key of* Professor*)*
Since DATE, -- *attribute*
Status CHAR (10), -- *attribute*
DeptId CHAR (4) NOT NULL, -- *role (key of* Department*)*
PRIMARY KEY (*ProfId*), -- *since a professor works in at most one department*
- FOREIGN KEY (*DeptId*) REFERENCES Department)

Entity or Attribute?

- Sometimes information can be represented as either an entity or an attribute.

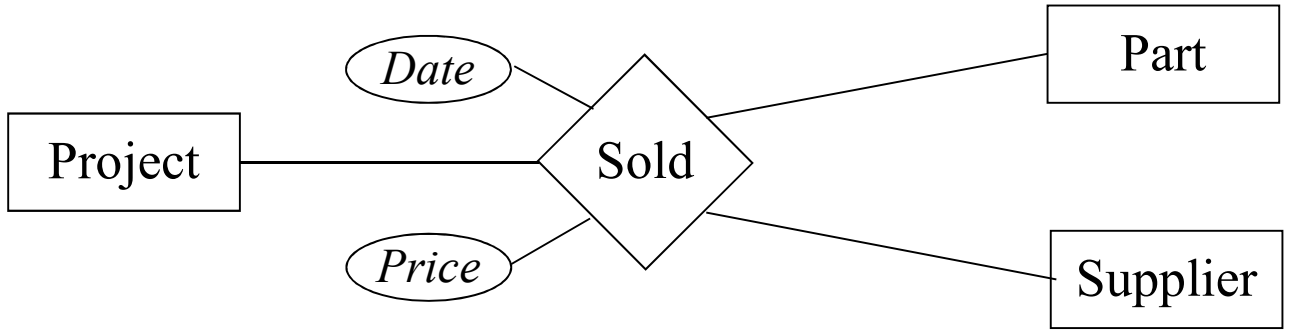


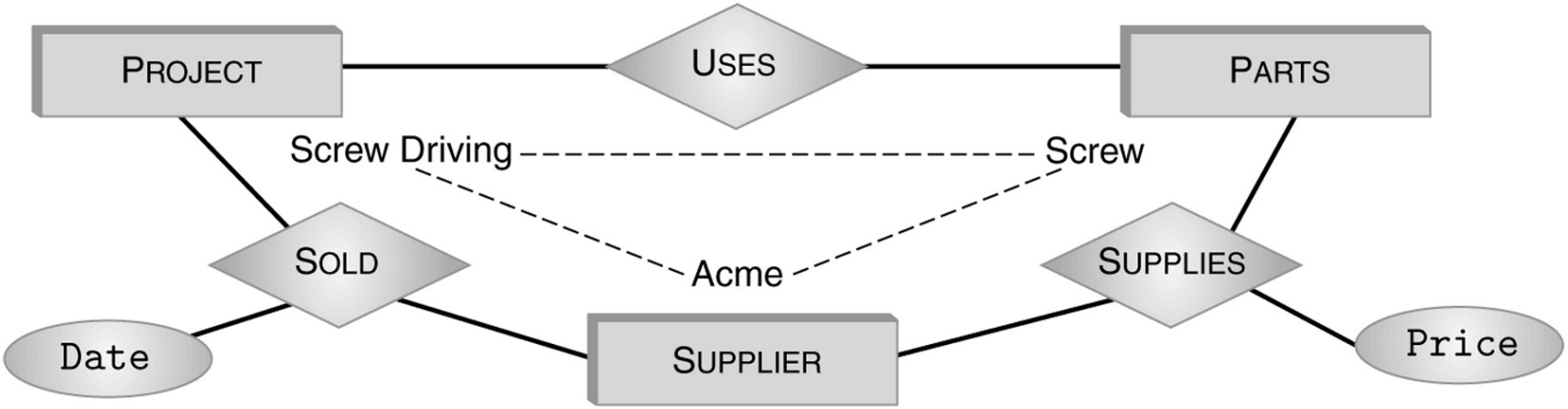
Entity or Relationship?



(Non-) Equivalence of Diagrams

- Transformations between binary and ternary relationships.





General Assertions

```
CREATE Assertion 3pc
CHECK NOT EXISTS (
        SELECT * FROM Papers P
        WHERE 3 <> (
            SELECT COUNT(*)
            FROM Review R
            WHERE R.paperid = P.paperid
         )
)
```

- But most DBMSs do not implement assertions
- Because it is hard to support them efficiently
- Instead, they provide **triggers**

Database Triggers

- Event-Condition-Action rules
- Event
  - Can be insertion, update, or deletion to a relation
- Condition
  - Can be expressed on DB state before or after event
- Action
  - Perform additional DB modifications

More About Triggers

- Row-level trigger
  - Executes once for each modified tuple

- Statement-level trigger
  - Executes once for all tuples that are modified in a SQL statement

Database Triggers Example

Product (name, price, category)

When Product.price is updated, if it is decreased then set Product.category = 'On sale'

```
CREATE TRIGGER ProductCategories
AFTER UPDATE OF price ON Product
REFERENCING
    OLD ROW AS OldTuple
    NEW ROW AS NewTuple
FOR EACH ROW
WHEN (OldTuple.price > NewTuple.price)
UPDATE Product
        SET category = 'On sale'
        WHERE productID = OldTuple.productID
```

What's SQL code look like in the real world?

- A lot of Stored Procedures
  - Can be executed by the trigger
  - More like a function or method in imperative languages (C++, Java, etc.)
  - There are fine-grained controls like
    - If-else check
    - While loop

Mapping natural language

ER components can be equated to parts of speech, as Peter Chen did. This shows how an ER Diagram compares to a grammar diagram:

- **Common noun:** Entity type. Example: student.
- **Proper noun:** Entity. Example: Sally Smith.
- **Verb:** Relationship type. Example: Enrolls. (Such as in a course, which would be another entity type.)
- **Adjective:** Attribute for entity. Example: sophomore.
- **Adverb:** Attribute for relationship. Example: digitally.

https://www.lucidchart.com/pages/er-diagrams

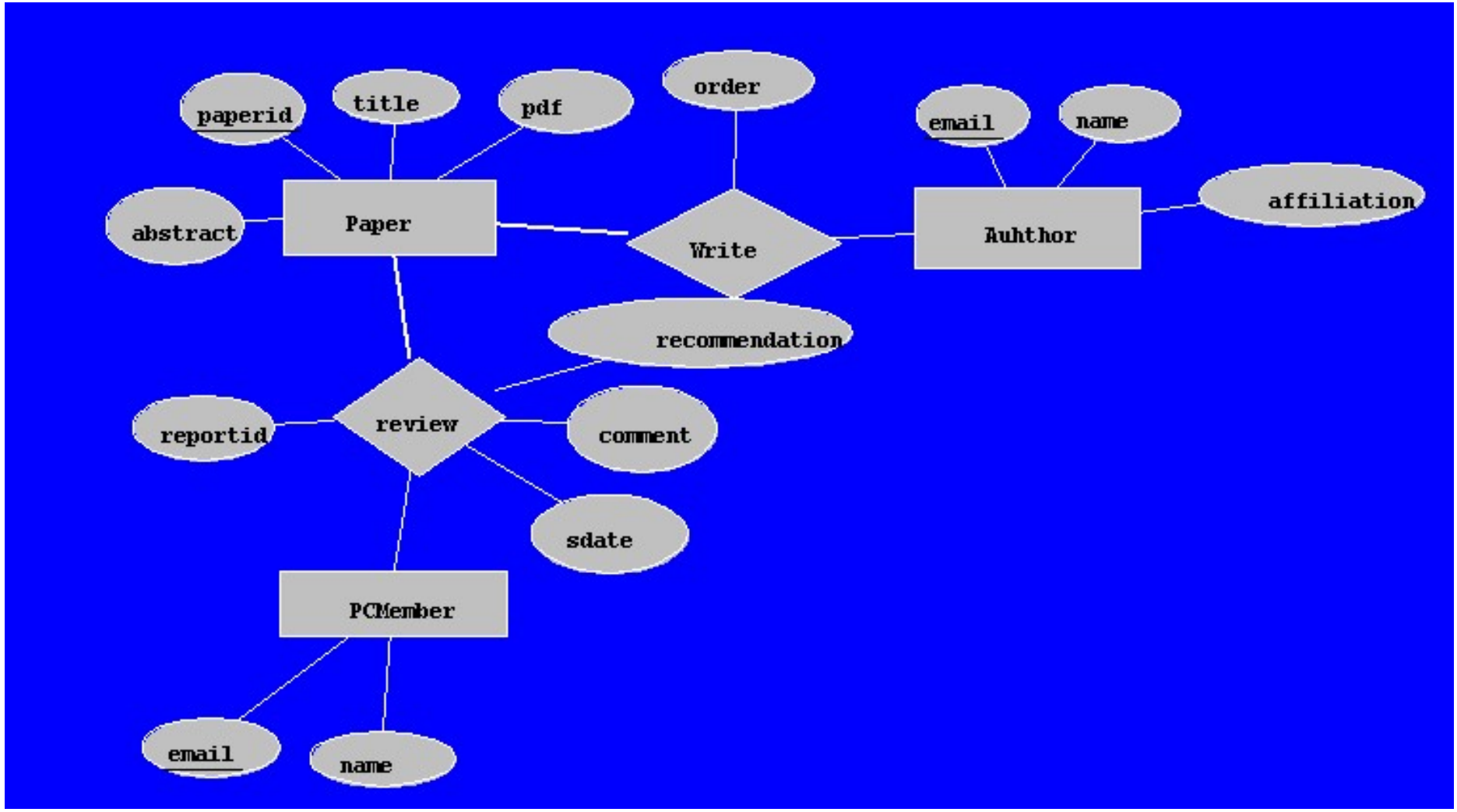
ER case study 1

Consider the design of the following database system for managing a conference X: a collection of papers are submitted to X, each of which has a unique paper IDs, a list of authors (names, affiliations, emails) in the order of contribution significance, title, abstract, and a PDF file for its content. The conference has a list of program committee (PC) members to review the papers. To ensure review quality, each paper is assigned to 3 PC members for review. To avoid overloading, each PC member is assigned with at most 5 papers, assuming that there are enough PC members. Each review report consists of a report ID, a description of review comment, a final recommendation (accept, reject), and the date the review report is submitted. A PC member can submit at most one review report for the paper that is assigned to him/her.

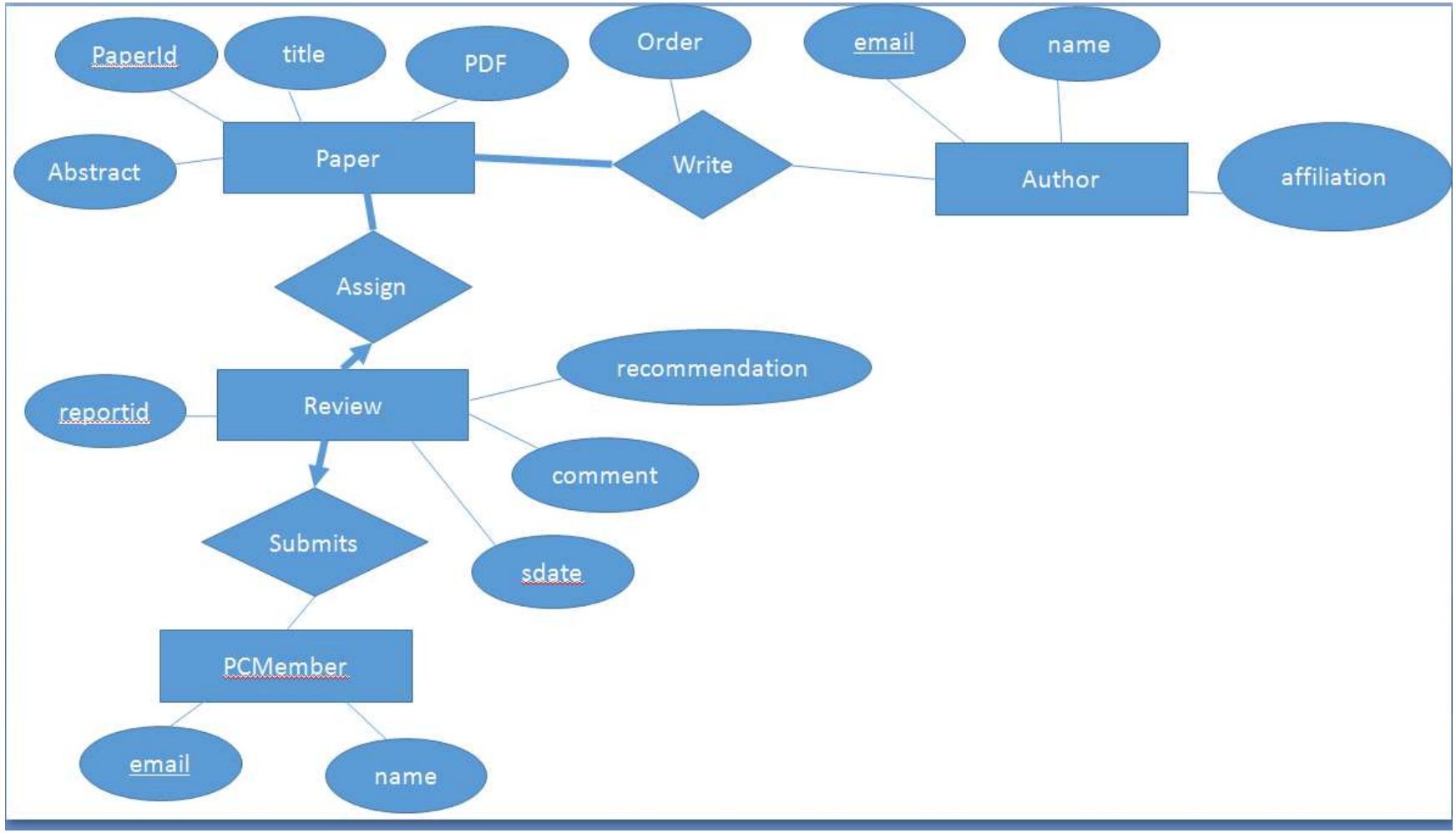
ER case study 1 (con't)

- Draw an E-R diagram for the above system. Use underlines, thick lines, and arrows to represent constraints. State your assumptions if necessary.
- Translate the previous E-R diagram for exercise1 into a relational model, i.e., a set of CREAT TABLE statements enforcing all stated constraints. In addition, write a CREATE ASSERTION statement to enforce that no PC member will be assigned to a paper of which she/he is a coauthor.

ER Diagram

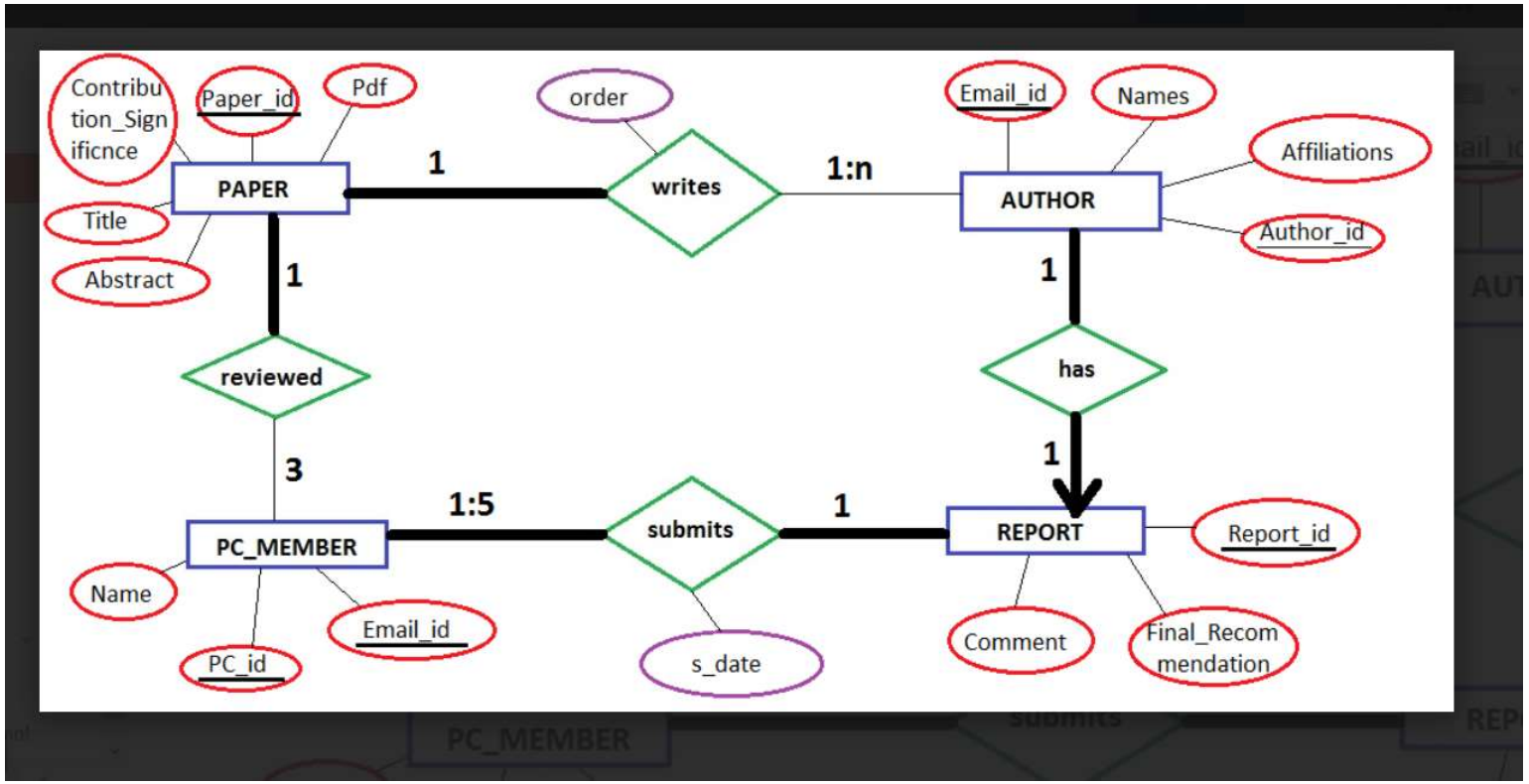

paperid
title
pdf
order
email
name
abstract
Paper
Write
Auhthor
affiliation
recommendation
review
reportid
comment
sdate
PCMember
email
name

How about modeling 'Review' as an entity type?



Credit of Renuka Gangireddy

Another notation, it correct yet?



Credit of Yukti Dhiman

SQL statements

```
Create table paper (
    paperid integer,
    title VARCHAR(50),
    abstract VARCHAR(250),
    pdf VARCHAR(100),
    primary key (paperid)
)
```

SQL statements

```
Create table author(
    email VARCHAR(100),
    name VARCHAR(50),
    affiliation VARCHAR(100),
    primary key(email)
)
```

```
CREATE table write(
    paperid integer,
    email varchar(50),
    order integer,
    Primary key(paperid, email),
    foreign key paperid references paper,
    foreign key email references autor
    )
```

```
create table pcmember(
    email VARCHAR(50),
    name VARCHAR(20),
    primary key (email)
)
```

```
create table review(
    reportid integer,
    sdate DATE,
    comment VARCHAR(250),
    recommendation CHAR(1),
    paperid integer,
    email VARCHAR(100),
    unique(paperid, email),
    foreign key paperid references paper,
    foreign key email references pcmember
    )
```

```
CREATE Assertion 3pc
CHECK NOT EXISTS (
        SELECT * FROM Papers P
        WHERE 3 <> (
            SELECT COUNT(*)
            FROM Review R
            WHERE R.paperid = P.paperid
         )
)
```

The “Abnormal” pattern:

Abnormal papers: those whose number of reviewers is not equal to 3.

```
CREATE ASSERTION atmostfivepapers
CHECK NOT EXISTS
(
        SELECT * FROM pcmember P
        WHERE 5 < (
              SELECT *
              FROM review R
              WHERE R.email = P.email
         )
)
```

The "Abnormal" pattern:

Abnormal pcmembers: those who review more than five papers.

ER case study 2

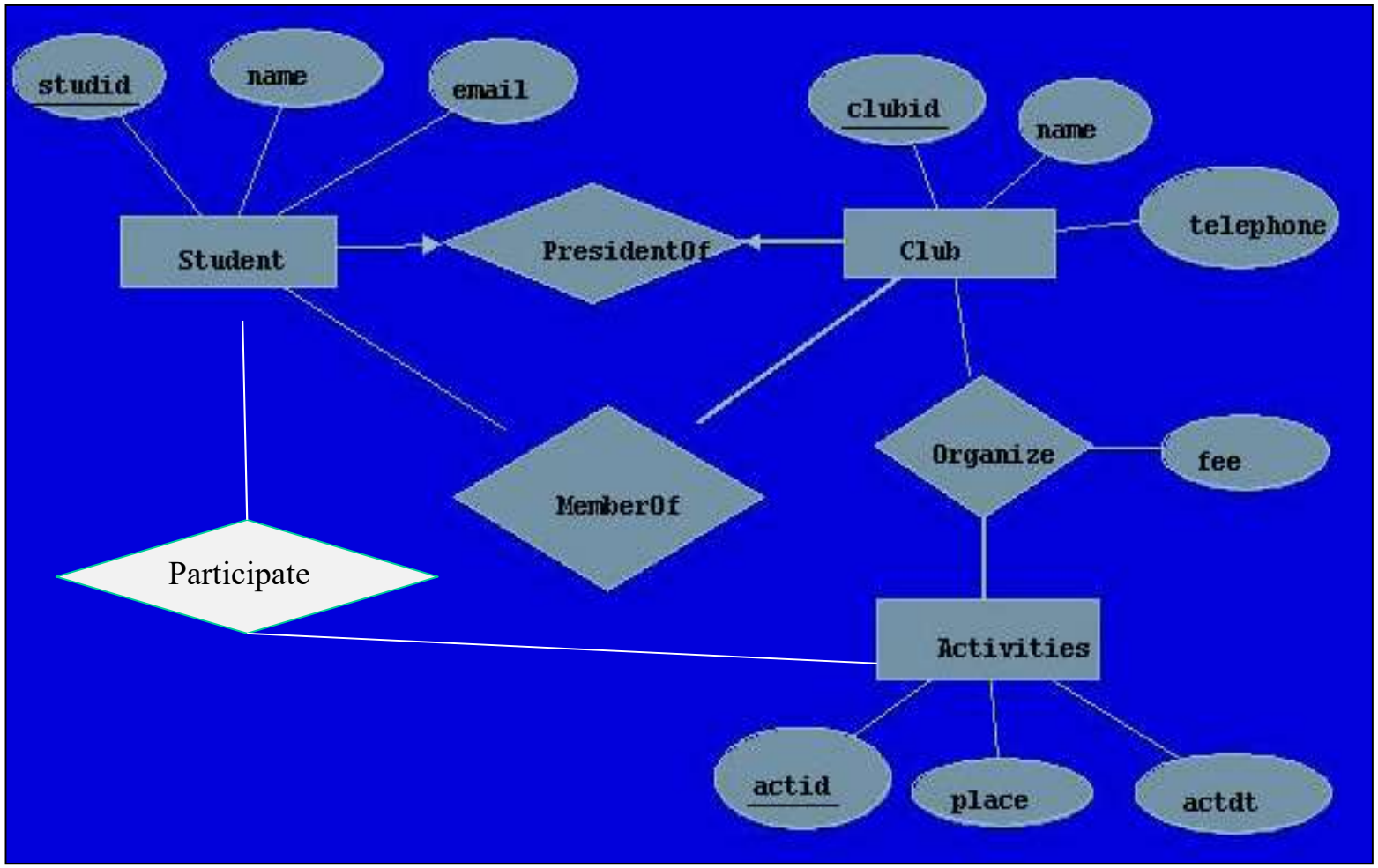
Suppose you are asked to design a club database system based on the following information.

Each student has a unique student id, a name, and an email; each club has a unique club id, a name, a contact telephone number, and has exactly one student as its president. Each student can serve as a president in at most one of the clubs, although he/she can be the members of several clubs. Clubs organize activities and students can participate in any of them. Each activity is described by a unique activity id, a place, a date, a time and those clubs that organize it. If an activity is organized by more than one club, different clubs might contribute different activity fees.

ER case study 2 (con't)

- Draw an E-R diagram for the system, in particular, use arrows or thick lines to represent constraints appropriately. Write down your assumptions if necessary.
- Translate the above E-R diagram to a relational model, in particular, specify your primary key and foreign key constraints clearly.

Reference solution



```
CREATE TABLE Student (
    studid CHAR(9),
    email VARCHAR(50),
    name VARCHAR(50) NOT NULL,
    PRIMARY KEY ( studid )
)
```

```
CREATE TABLE Club (
    clubid INTEGER,
    telephone VARCHAR(15),
    name VARCHAR(50),
    president CHAR(9) NOT NULL,
    unique(president),
    PRIMARY KEY (clubid),
    FOREIGN KEY (president) REFERENCES Student(studid )
)
```

```
CREATE TABLE MemberOf (
    clubid INTEGER,
    studid VARCHAR(50),
    PRIMARY KEY (clubid, studid ),
    FOREIGN KEY (clubid ) REFERENCES Club( clubid ),
    FOREIGN KEY (studid ) REFERENCES Student( studid )
)
```

```
CREATE TABLE Activities (
    actid INTEGER,
    actdt DATETIME,
    place VARCHAR(50),
    PRIMARY KEY (actid)
)
```

```
CREATE TABLE Organize (
    actid INTEGER,
    clubid INTEGER,
    fee VARCHAR(50),
    PRIMARY KEY (actid, clubid),
    FOREIGN KEY (actid ) REFERENCES Activities(actid),
    FOREIGN KEY (clubid ) REFERENCES Club(clubid)
)
```

```
CREATE ASSERTION AtLeastOneOrganizer
CHECK NOT EXISTS(
        SELECT *
        FROM Activities
        WHERE NOT EXISTS(
                SELECT *
                FROM Organize
                WHERE Organize.actid=Activities.actid
        )
)
```

The “Abnormal” pattern:

Abnormal activities: those that have no organizers

```
CREATE ASSERTION AtLeastOneOrganizer
CHECK NOT EXISTS(
        SELECT *
        FROM Activities
        WHERE 0 =  (
                SELECT COUNT(*)
                FROM Organize
                WHERE Organize.actid=Activities.actid
        )
)
```

The “Abnormal” pattern:

Abnormal activities: those that have no organizers

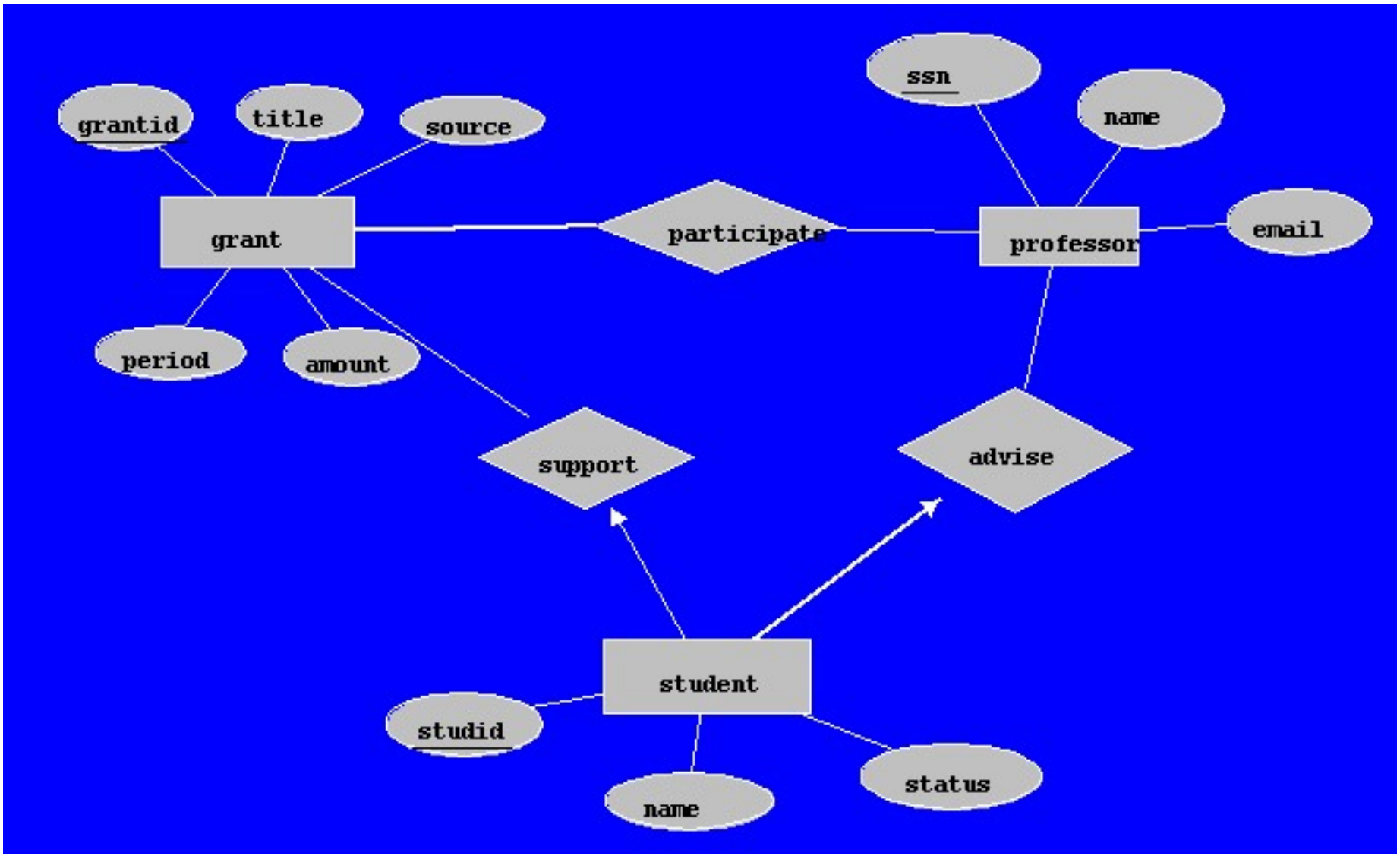
ER case study 3

Consider the design of a database for the management of grants. Each grant is identified by a unique grant ID, a title, the funding source of the grant, the period (starting date and ending date), and the amount of grant. Each grant might be participated by several professors and each professor might also participate in several grants. Each professor is identified by a unique SSN, name, and email address. In addition, several graduate students might be supported by a grant as GRAs, although each student can be supported by at most one grant. Each graduate student has exactly one professor as his/her advisor.

ER case study 3 (con't)

- Draw an E-R diagram for the system, in particular, use arrows or thick lines to represent constraints appropriately. Write down your assumptions and justifications briefly and clearly.
- Translate the above E-R diagram into a relational model, i.e., write a set of CREATE TABLE statements. In particular, specify primary key, foreign key and other constraints whenever possible.

Reference solution



```
create table grant(
        grantid integer,
        title varchar(50),
        source varchar(50),
        periodstart DATE,
        periodend DATE,
        amount integer,
        primary key(grantid)
)
```

```
Create table professor(
    ssn char(9),
    name VARCHAR(20),
    email varchar(20),
    primary key(ssn),
    unique(email)
)
```

```
Create table participate(
    grantid integer,
    professorid char(9),
    primary key(grantid, professorid),
    foreign key grantid references grant,
    foreign key professorid references professor(ssn)
    )
```

```
Create student(
    studid integer,
    name varchar(50),
    status varchar(20),
    advisor char(9) NOT NULL,
    supportgrantid integer,
    primary key(studid),
    foreign key advisor references professor,
    Foreign key supportgrantid references grant(grantid)
)
```

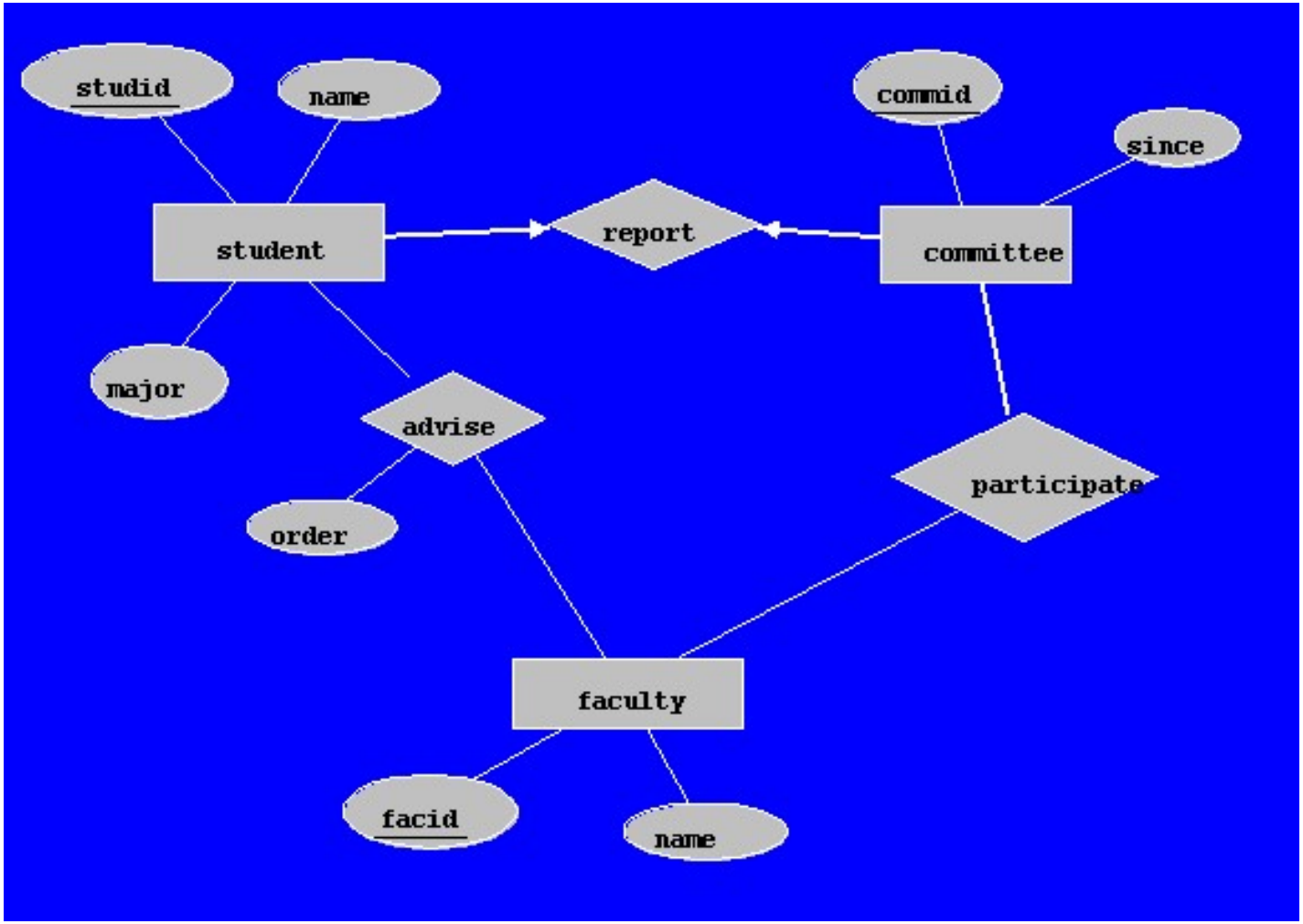
ER case study 4

Consider the design of the following database system: each PhD student has exactly one dissertation committee which consists of 4-5 faculty, and each committee is for exactly one student. Each student has an ordered list of advisors including the primary advisor followed by 0 or more secondary advisors. Each student has a unique studid, a name, and a major. Each committee has a unique committee id, and the date the committee is formed. Each faculty has a unique facid and a name. Each faculty can participate in multiple committees and be the advisors (either primary or secondary) of several students.

ER case study 4 (con't)

- Draw an E-R diagram for the above system. Use underlines, thick lines, and arrows to represent constraints. State your assumptions if necessary.
- Translate your E-R diagram for problem 1 into a relational model, i.e., a set of CREAT TABLE/ASSERTION statements enforcing all stated constraints. In addition, write a CREATE ASSERTION statement to enforce that each committee consists of the primary advisor of the student and all other members of the committee cannot be the secondary advisors of the student.

Reference solution



Reference solution

```
CREATE TABLE Advise (
    studid CHAR(9),
    facid VARCHAR(50),
    order INTEGER,
    PRIMARY KEY ( studid, facid),
    UNIQUE(studid, order),
    FOREIGN KEY ( studid ) REFERENCES Students (
    studid ),
    FOREIGN KEY (facid ) REFERENCES Faculty ( facid )
)
```

```
CREATE TABLE Student (
    studid CHAR(9) NOT NULL,
    name VARCHAR(50),
    major VARCHAR(50),
     since DATE,
    PRIMARY KEY ( studid )
)
```

```
CREATE TABLE Participate (
    studid CHAR(9),
    facid CHAR(9),
    PRIMARY KEY (studid, facid ),
    FOREIGN KEY ( studid ) REFERENCES Student
    FOREIGN KEY (facid ) REFERENCES Faculty ( facid )
)
```

The primary advisor must be in the committee

```
CREATE ASSERTION
CHECK NOT EXISTS(
        SELECT * from Student S
        WHERE (
                SELECT facid            // one primary advisor
                FROM Advise A
                WHERE A.stuid = S.stuid  and A.order = 1
        ) NOT IN
        (                       // all my committee members
                select facid
                FROM Participate P
                WHERE P.stuid = S.studid
        )
)
```

Other co-advisors must be NOT in the committee

```
CREATE ASSERTION
CHECK NOT EXISTS(
        SELECT * from Student S
        WHERE EXISTS ( // some committee members are co-advisors
                SELECT A.facid
                FROM Advise A, Pariticpate P
                WHERE A.stuid = S.stuid
                AND A.stuid = P.stuid
                AND A.order <> 1
                AND A.facid = P.facid
)
```