CPSC 304 – Introduction to Relational Databases – Summer 2004

Midterm Examination - Key for all questions

July 8, 2004

NAME: ____

Student ID: ________________________________

Instructions

1. This is a **closed book, closed notes** exam.
2. The exam is for two hours: from 12 : 45 – 2 : 45 P.M.
3. Your answer should be your own work. Any plagiarism will result in a **zero** in the exam.
4. Be clear and precise in your work. Check your answers before turning them in.
5. Ask the instructor for work sheets in case you need to use them for rough work. Attach them if necessary. Put your name and student id on each of the worksheets you attach.

POINTS

Prob.	Points	Max. Points
1(a)		15
1(b)		35
2(a)		35
2(b)		5
3(a)		10
3(b)		10
3(c)		10

Prob.	Points	Max. Points
3(d)		10
3(e)		10
4(a)		5
4(b)		5
4(c)		5
4(d)		5
4(e)		5

TOTAL: ________________________ **OUT OF:** **165 points**

Answer all questions.

1. Hogwarts school of witchcraft and wizardry has decided to step into the modern world and has asked you to design its relational database. It has given the following list of basic conditions that are needed for its database.

   - Professors have a Wizard ID which is a 10 digit number, a name, a rank, the number of years they have been in the school and their research specialty.
   - Students have an Apprentice ID which is a 10 digit number, a name, the year of study and their gpa.
   - Professors teach Courses which are offered by the school. A course has a course ID, a name and the number of credits. Every course is taught by atmost one professor and a professor may teach many courses. Courses are taught in *terms*. The teacher of a course may vary for different terms.
   - There are various *houses* to which students belong to. A student is a member of exactly one house.
   - Houses are identified by their house ID and have a name and the number of students in the school who are currently a member of the house. In addition, every house has precisely one student who is the *head* of the house. Each house name is distinct.
   - Students are also enrolled in courses. A student can enroll in many courses in a given term and a course contains many students. The grade of the student in the course that is offered in a term is also to be noted along with the enrollment information and is unique. A student may enroll in the same course in different terms.
   - Every student has atmost one professor who advises the student. A professor may advise MANY students.

   (a) List the various entities and their corresponding attributes in the design. For each relationship, describe the key constraints, type of the relationship and descriptive attributes (if any).

   Ans: The **entities** and their attributes are:

   i. **Professor**: Wizard ID (WID), Number of Years (numYears), Professor Name (pname), Rank (rank) and Specialization (spec)
   ii. **Student**: Apprentice ID (SID), Student Name (sname), Year of Study (year), gpa (gpa)
   iii. **Course**: Course ID (CID), Course Name (cname), Number of Credits (numCred)
   iv. **House**: House ID (HID), House Name (hname), Number of Members (numMem), Head ID (headID)

   The relationships are:

   i. **Teaches**: **Key constraint** on Course - A course can have at most one professor who teaches it, Relationship is one to many - A professor can teach many courses, Descriptive attribute of the relationship is the Term of offering (term)
   ii. **Advises**: **Key constraint** on advises - A student can have at most one professor who advises the student, Relationship is one to many - A professor can advise many students, There are no descriptive attributes

iii. **Enrolled**: The relationship is many to many, The Descriptive attributes are grade and term of offering. Additionally there is a foreign key constraint that (cid,term) is a foreign key in the relation Teaches. This models the scenario that a student is enrolled in a course that is taught or offered in a given term. (However, I have not strictly enforced this constraint for grading)

iv. **MemberOf**: **Key constraint** on MemberOf - A student can be a member of at most one house, Relationship is one to many - A house can have many students, There are no descriptive attributes.

(b) Design and draw an ER diagram that captures the information required by the school. Use entities, relationships and attributes and be sure to indicate the key and participation constraints.

Ans: The ER diagram is shown in figure 1. There are a few points to note. The advises relationship has *total* participation from students. However, since the question says *at-most* one professor who advises a student, there is some ambiguity in interpretation here. So an answer where the thick line is absent for the advises relationship from students is also a valid one.

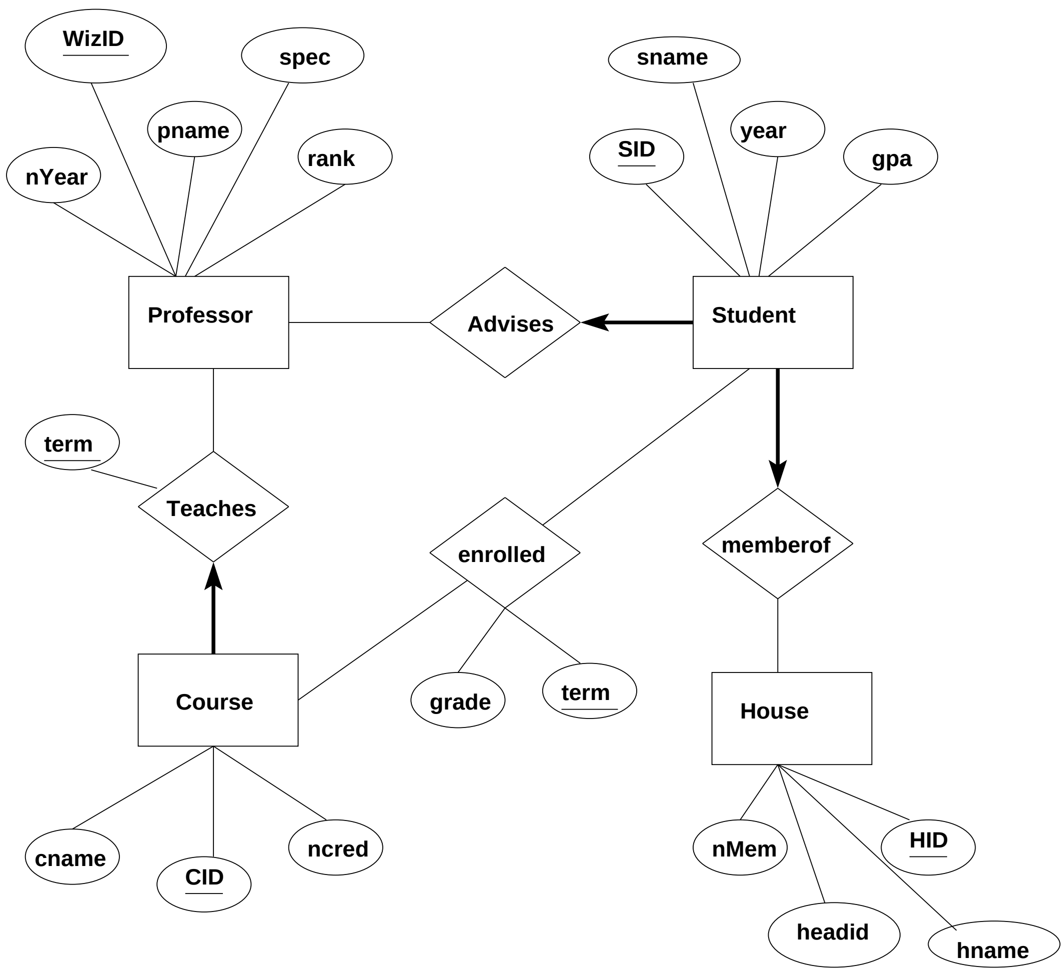

Figure 1: ER-Model

2. Consider the ER diagram you designed for the previous question.

   (a) List the relations for the ER diagram of Hogwarts. Also give SQL statements to create the tables for the relations. Specify only the PRIMARY and FOREIGN KEY constraints in the SQL statements.

   Ans: The following are the relations.

      i. Professor(<u>WizID:integer</u>, pname:string, nYear:integer, spec:string, rank:string)
      ii. Student(<u>SID:integer</u>, sname:string, year:integer, gpa:real)
      iii. Course(<u>CID:integer</u>, cname:string, ncred:integer)
      iv. House(<u>HID:integer</u>, hname:string, nMem:integer, headid:integer)
      v. Advises(<u>WizID:integer, SID:integer</u>)
      vi. Enrolled(<u>CID:integer, SID:interger, term:integer</u>, grade:char)
      vii. Teaches(<u>WizID:integer, CID:integer, term:string</u>)
      viii. MemberOf(<u>SID:integer, HID:integer</u>)

   The SQL statements to create the tables are given below.

```
CREATE TABLE Professor ( WizID    INTEGER,
                             pname    CHAR(30),
                             nYear     INTEGER,
                             spec      CHAR(30),
                             rank      CHAR(10),
                             PRIMARY KEY (WizID) )

CREATE TABLE Student ( SID     INTEGER,
                           sname   CHAR(30),
                           year    INTEGER,
                           gpa     REAL,
                           PRIMARY KEY (SID) )

CREATE TABLE Course ( CID   INTEGER,
                          cname   CHAR(30),
                          ncred    INTEGER,
                          PRIMARY KEY (CID) )

CREATE TABLE House ( HID   INTEGER,
                          hname  CHAR(30),
                          nMem   INTEGER,
                          headid    INTEGER,
                          UNIQUE (hname),
                          PRIMARY KEY (HID)
                          FOREIGN KEY (headid) REFERENCES Student (SID) )

CREATE TABLE Advises ( WizID   INTEGER,
                           SID     INTEGER,
                           PRIMARY KEY (WizID, SID),
                           FOREIGN KEY (WizID)  REFERENCES Professor,
                           FOREIGN KEY (SID)  REFERENCES Student )
```

```
CREATE TABLE Teaches ( WizID   INTEGER,
                            CID      INTEGER,
                            term     CHAR(20),
                            PRIMARY KEY (CID, term),
                            FOREIGN KEY (WizID)  REFERENCES Professor,
                            FOREIGN KEY (CID)  REFERENCES Course )

CREATE TABLE Enrolled ( SID      INTEGER,
                            CID      INTEGER,
                            term     CHAR(20),
                            grade    CHAR,
                            PRIMARY KEY (SID, CID, term),
                            FOREIGN KEY (SID)  REFERENCES Student,
                            FOREIGN KEY (CID)  REFERENCES Course,
                            FOREIGN KEY (CID,term)  REFERENCES Teaches )

*** Note that the last constraint was not taken into account for grading purposes ***


CREATE TABLE MemberOf (  SID   INTEGER,
                               HID  INTEGER,
                               PRIMARY KEY (SID, HID),
                               FOREIGN KEY (SID)  REFERENCES Student,
                               FOREIGN KEY (HID)  REFERENCES House )
```

(b) Suppose it is required that the Enrolled relationship has to take care of *foreign key violations* with respect to students and courses in the following manner. When a student is removed from the school, delete all the enrolled rows that refer to the students row. The same happens in the case of an update. How should the SQL statement to create the Enrolled table be changed to reflect this requirement.

Ans: The SQL statement is as below.

```
CREATE TABLE Enrolled ( SID      INTEGER,
                        CID      INTEGER,
                        term     CHAR(20),
                        grade    CHAR,
                        PRIMARY KEY (SID, CID, term),
                        FOREIGN KEY (CID)  REFERENCES Course,
                        FOREIGN KEY (SID)  REFERENCES Student
                             ON DELETE CASCADE
                             ON UPDATE CASCADE )
```

3. Consider the following schema.

Suppliers(*sid*:integer, *sname*:string, *address*:string)
Parts(*pid*:integer, *pname*:string, *color*:string)
Catalog(*sid*:integer, *pid*:string, *cost*:real)

The key fields of each relation are underlined. Write the following queries in Relational Algebra and SQL. (There may be many ways to answer each query.)

(a) Find the *names* of suppliers who supply some *yellow* part.

RA: $\pi_{sname}(\text{Suppliers} \bowtie (\sigma_{color='yellow'}(\text{Parts}) \bowtie \text{Catalog}))$

SQL:

```
SELECT   S.sname
FROM     Suppliers S, Parts P, Catalog C
WHERE   S.sid = C.sid AND P.pid = C.pid AND P.color = 'yellow'
```

(b) Find the *sids* of suppliers who supply some *green* part but not a *red* part.

RA: $\pi_{sid}(\text{Suppliers} \bowtie (\sigma_{color='green'}(\text{Parts}) \bowtie \text{Catalog})) -$
$\pi_{sid}(\text{Suppliers} \bowtie (\sigma_{color='red'}(\text{Parts}) \bowtie \text{Catalog}))$

SQL:

```
SELECT   S.sid
FROM     Suppliers S, Parts P, Catalog C
WHERE   S.sid = C.sid AND P.pid = C.pid AND P.color = 'green'
EXCEPT
SELECT   T.sid
FROM    Suppliers T, Parts P, Catalog C
WHERE   T.sid = C.sid AND P.pid = C.pid AND P.color = 'red'
```

(c) Find the *sids* and *snames* of suppliers who supply a *'bolt'* whose price is under 100 dollars or whose color is *red*.

There is ambiguity here too. Some of you may interpret the question to mean a bolt whose price is under 100 dollars or a bolt whose color is red. Some of you may interpret it to mean a bolt whose price is under 100 dollars or a red part. I have given points to both these versions.

RA: $\pi_{sid,sname}(\text{Suppliers} \bowtie (\sigma_{(pname='bolt' \wedge cost<100) \vee (color='red')}(\text{Parts} \bowtie \text{Catalog})))$

RA: $\pi_{sid,sname}(\text{Suppliers} \bowtie (\sigma_{(pname='bolt' \wedge (cost<100 \vee color='red'))}(\text{Parts} \bowtie \text{Catalog})))$

SQL:

```
SELECT   S.sid, S.sname
FROM    Suppliers S, Parts P, Catalog C
WHERE  S.sid = C.sid AND P.pid = C.pid AND P.pname = 'bolt' AND C.cost < 100
UNION
SELECT   T.sid, T.sname
FROM    Suppliers T, Parts P, Catalog C
WHERE  T.sid = C.sid AND P.pid = C.pid AND P.color = 'red'
```

```
SELECT   S.sid, S.sname
FROM    Suppliers S, Parts P, Catalog C
WHERE  S.sid = C.sid AND P.pid = C.pid AND P.pname = 'bolt' AND C.cost < 100
UNION
SELECT   T.sid, T.sname
FROM    Suppliers T, Parts P, Catalog C
WHERE  T.sid = C.sid AND P.pid = C.pid AND P.pname = 'bolt' AND P.color = 'red'
```

(d) Find the *sids* of suppliers who supply all parts.
RA: $\pi_{sid}((\pi_{sid,pid}(Catalog))/(\pi_{pid}(Parts))$
SQL:

```
SELECT  C.sid
FROM   Catalog C
WHERE NOT EXISTS (( SELECT P.pid
                          FROM  Parts P )
                          EXCEPT
                          (SELECT C1.pid
                           FROM  Catalog C1
                           WHERE C1.sid = C.sid ))
```

(e) Find *pids* of parts supplied by at least two different suppliers
RA: $\rho(CatPairs(1 \rightarrow sid_1, 2 \rightarrow pid_1, 3 \rightarrow cost_1, 4 \rightarrow sid_2, 5 \rightarrow pid_2, 6 \rightarrow cost_2), Catalog \times Catalog)$
$\pi_{pid_1}(\sigma_{(pid_1 = pid_2) \wedge (sid_1 \neq sid_2)} CatPairs)$
SQL:

```
SELECT   P.pid
FROM    Catalog P, Catalog C
WHERE  P.pid = C.pid AND P.sid <> C.sid
```

(f) Find the *names* of suppliers who supply some *brown* part.
RA: $\pi_{sname}(Suppliers \bowtie (\sigma_{color='brown'}(Parts) \bowtie Catalog))$
SQL:

```
SELECT   S.sname
FROM     Suppliers S, Parts P, Catalog C
WHERE   S.sid = C.sid AND P.pid = C.pid AND P.color = 'brown'
```

(g) Find the *sids* of suppliers who supply some *yellow* part but not a *red* part.
RA: $\pi_{sid}(Suppliers \bowtie (\sigma_{color='yellow'}(Parts) \bowtie Catalog)) -$
$\pi_{sid}(Suppliers \bowtie (\sigma_{color='red'}(Parts) \bowtie Catalog))$
SQL:

```
SELECT   S.sid
FROM     Suppliers S, Parts P, Catalog C
WHERE   S.sid = C.sid AND P.pid = C.pid AND P.color = 'yellow'
EXCEPT
SELECT   T.sid
FROM    Suppliers T, Parts P, Catalog C
WHERE   T.sid = C.sid AND P.pid = C.pid AND P.color = 'red'
```

(h) Find the *sids* and *snames* of suppliers who supply a *'nut'* whose price is under 10 dollars or whose color is *pink*.

There is ambiguity here too. Some of you interpret the question as a nut whose price is under 10 dollars or a nut whose color is pink. Some of you interpret it as a nut whose price is under 10 dollars and a part whose color is pink. Both versions are treated as correct answers.

RA: $\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='nut' \wedge cost<10) \vee (color='pink')}(Parts \bowtie Catalog)))$

RA: $\pi_{sid,sname}(Suppliers \bowtie (\sigma_{(pname='nut' \wedge (cost<10 \vee color='pink'))}(Parts \bowtie Catalog)))$

SQL:

```
SELECT   S.sid, S.sname
FROM     Suppliers S, Parts P, Catalog C
WHERE  S.sid = C.sid AND P.pid = C.pid AND P.pname = 'nut' AND C.cost < 10
UNION
SELECT   T.sid, T.sname
FROM     Suppliers T, Parts P, Catalog C
WHERE  T.sid = C.sid AND P.pid = C.pid AND P.color = 'pink'
```

```
SELECT   S.sid, S.sname
FROM     Suppliers S, Parts P, Catalog C
WHERE  S.sid = C.sid AND P.pid = C.pid AND P.pname = 'nut' AND C.cost < 10
UNION
SELECT   T.sid, T.sname
FROM     Suppliers T, Parts P, Catalog C
WHERE  T.sid = C.sid AND P.pid = C.pid AND P.pname = 'nut' AND P.color = 'pink'
```

4. Consider the following instances over the schema given below.

   Sailors(*sid:integer*, *sname:string*, *rating:integer*, *age:real*)

   Boats(*bid:integer*, *bname:string*, *color:string*)

   Reserves(*sid:integer*, *bid:integer*, *day:date*)

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Show the result of the following queries on the instances of the relation schema given above.

(a) Query 1:

```
SELECT  S.sname
FROM    Sailors S
WHERE   S.sid NOT IN ( SELECT  R.sid
                           FROM    Reserves R
                           WHERE   R.bid = 103 )
```

The answer is

sname
Brutus
Andy
Rusty
Horatio
Zorba
Art
Bib

(b) Query 2:

```
SELECT   S.sid
FROM     Sailors S
WHERE   S.rating > ANY ( SELECT  S2.rating
                             FROM    Sailors S2
                             WHERE   S2.sname = 'Horatio' )
```

The answer is

sid
31
32
58
71
74

(c) Query 3:

```
SELECT  S.sname
FROM   Sailors S
WHERE  NOT EXISTS  (( SELECT  B.bid
                            FROM   Boats B )
                            EXCEPT
                            (SELECT  R.bid
                             FROM   Reserves R
                             WHERE  R.sid = S.sid ))
```

The answer is

sname
Dustin

(d) Query 4:

```
SELECT  S.sname
FROM   Sailors S
WHERE  S.age > ( SELECT  MAX ( S2.age )
                     FROM   Sailors S2
                     WHERE S2.rating = 10 )
```

The answer is

sname
Dustin
Lubber
Bob

(e) Query 5:

```
SELECT   S.sid, S.sname
FROM    Sailors S
WHERE  S.rating > ALL ( SELECT  S2.rating
                            FROM   Sailors S2
                            WHERE  S2.sname = 'Horatio' )
```

The answer is

sid	sname
58	Rusty
71	Zorba