School of Science

ISYS1055/1057 Database Concepts

Assignment 2 solution and marking guide

Question 1. SQL (15 points).

Each question is worth 1 point

NOTE: where individual questions state requirements regarding operators, these must be followed. Otherwise, **equivalent** variations that produce the correct answer are acceptable (e.g. using a subquery instead of a join; using a different join operators instead of an explicit equality condition join; etc)

1. Explain the query below in English. Literal explanation will receive 0 mark.

```
select givename, famname, instname
from academic natural join department
where acnum in
      (select acnum
      from author
      where acnum not in
            (select acnum
            from interest
            group by acnum))
and   deptNum in
        (select deptNum
         from academic
         where deptname = ‘Computer Science’);
```

English explanation: Find the first and last name of academics and the name of the institution they work for if they have authored (or co-authored) any papers but do not have any interests and work in a Computer Science department.

2. The following SQL query is meant to output a list of papers (panum) with the total number of authors for each paper. It has syntax errors and logic errors. Give the correct query.

```
select PaNum, count(A1.AcNum)
from Author A1, Author A2
where PaNum = A2.PaNum
group by PaNum;
```

Half-correct SQL query (half mark) - the join is unnecessary and therefore the count(.) returns wrong result:

```
select A1.PaNum, count(A1.AcNum)
from Author A1, Author A2
where A1.PaNum = A2.PaNum
group by A1.PaNum;
```

Fully-correct SQL query - remove the Join:

```
select Panum, count(acnum)
        from Author
        group by PaNum;
```

3. Find departments that have description (descrip) is available in the database. Return all details of these departments.

```
select *
from department
where descrip is not null;
```

4. List the paper number and title of papers by the academic whose acnum is 100.

```
select paper.panum, title
from paper, author
where paper.panum=author.panum
  and author.acnum=100;
```

5. For each academic, give the acnum, givename, famname and the total number of papers s/he has written, Note that if an academic has not written any paper, his/her total should be zero.
You can use JOIN operators such as NATURAL, JOIN ...ON.

```
Select academic.acnum, givename, famname, count(panum)
from Academic left outer join author on academic.acnum=author.acnum
group by academic.acnum, givename, famname;

/* Alternative answer using set operator */
Select academic.acnum, givename, famname, count(panum)
from Academic join author on academic.acnum=author.acnum
group by academic.acnum, givename, famname
UNION
Select acnum, givename, famname, 0
from Academic
Where acnum not in (select acnum
                    From author);
```

6. The research field ID is a research field classification code representing classes for three "Levels". These three Levels are separated by a "full stop" in a single string. For example the research field ID "B.1.6" represents that the research field belongs to Class "B" for Level one, Class "1" for Level two and Class "6" for Level three. For research fields in Class "1" for Level two, list the field IDs and the number of academics for each field ID.

```
-- NOTE: string pattern match variants are also acceptable:
-- like '%.1.%' , like '%.1%', like '_.1%'

select field.ID, count(acnum)
from interest, field
where trim(field.ID) like '_.1._' and interest.fieldnum = field.fieldnum
group by field.ID;
```

7. Department where at least one academic does not have research interest
    List deptnum, depntname, instname of these departments.
    Must use a SUBQUERY.

```
-- Answer:
-- Using NOT EXISTS
select DISTINCT d.deptnum, deptname, instname
from department d, academic a
where d.DEPTNUM = a.DEPTNUM
```

```
	and NOT EXISTS (SELECT *
			FROM interest
			WHERE acnum = a.acnum);

-- Using NOT IN
select DISTINCT d.deptnum, deptname, instname
from department d, academic a
where d.DEPTNUM = a.DEPTNUM
	and a.acnum NOT IN (SELECT acnum
			    FROM interest);
```

8. Output in alphabetical order the acnum, famname, givename, department number (deptnum), and description (descrip) of authors whose family name starts with “C”.

```
select distinct academic.acnum, FamName, GiveName, academic.deptnum, descrip
from department, academic, author
where department.DeptNum=Academic.DeptNum
	and Academic.AcNum = Author.AcNum
	and FamName like 'C%'
order by Famname, givename, academic.deptnum, descrip;

/* Note: required ORDER BY ordering not specified in question, but should at least include famname (first) and givename (second); also having deptnum and descript is optional */

/* Note academic NATURAL JOIN author NATURAL JOIN department also works here */

/* Sub-queries can also be used */
select distinct academic.acnum, FamName, GiveName, academic.deptnum, descrip
from department, academic
where department.DeptNum=Academic.DeptNum
	and FamName like 'C%'
	and Academic.AcNum IN (select acnum from author)
order by Famname, givename, academic.deptnum, descrip;
```

9. List the fieldnum, title, and total number of interested academics (under the heading "NO. ACADEMICS INTERESTED") in each research field in increasing order (i.e. ascending order) of fieldnum.

```
-- Answer: (note: fields with zero interested academics do not need to be returned)
select f.fieldnum, title, count(i.acnum) as "NO. ACADEMICS INTERESTED"
from field f, interest i
where f.FIELDNUM = i.FIELDNUM
group by f.fieldnum, title
order by f.fieldnum;
```

10. List in alphabetical order the institution and name of departments where at least 10 academics having written papers.

```
select InstName, deptName
from department, academic, author
where department.deptnum=academic.deptnum
	and academic.acnum=author.acnum
group by department.deptnum, DeptName, Instname
having count(distinct author.acnum) >=10
```

```
order by instname, deptname;

/* Notes:
 1. distinct is necessary for counting correctly.
 2. group by deptName, Instname is necessary.*/
```

11. List the deptnum of departments whose postcodes are in the range 3000..3999 and that do not have any academics with the title of professor (stored as “Prof” or “Prof.” in the database) , including departments that do not have any academics.

```
select deptnum
from Department
where postcode >= 3000
   and postcode <=3999
minus
select deptnum
from Academic
where title like 'Prof%';
```

12. Find the departments whose academics have written at least ten papers. Output their deptnum and deptname in ascending order.

```
select D.deptnum, D.deptname
from DEPARTMENT D, ACADEMIC A, AUTHOR AU
where D.deptnum=A.deptnum
   and A.acnum=AU.acnum
group by D.deptnum, D.deptname
having count(distinct panum)>=10
order by D.deptnum, D.deptname;
```

1.13. List the deptnum and deptname of departments whose academics have never written any papers.

```
-- the first join is to ensure departments with academics

select deptnum, deptname
from DEPARTMENT D, ACADEMIC A
where D.deptnum=A.deptnum
minus
select deptnum, deptname
from DEPARTMENT D, ACADEMIC A
where D.deptnum=A.deptnum
   and acnum in (select acnum
            from AUTHOR);

/* Note: 1.13 can also be solved using subqueries, that’s also fine as an answer */
```

1.14. List papers (panum) by academics with research interests in fields related to "Data". You must use EXISTS. Note that “fields related to data” includes any occurrence of the four letters “data” within a field name, in any case.

```
select distinct panum
from AUTHOR
where exists (select *
   from INTEREST I, FIELD F
```

```
    where upper(F.title) like '%DATA%'
        and I.acnum=AUTHOR.acnum
        and I.fieldnum=F.fieldnum);
```

1.15. The popularity of a field is measured by the number of interested academics. List the details (fieldnum, id and title) of the most popular field together with the total number of interested academics.

```
select F.fieldnum, F.id, F.title, count(acnum)
from FIELD F, INTEREST I
where F.fieldnum=I.fieldnum
group by F.fieldnum, F.id, F.title
having count(acnum) >= all (select count(acnum)
                            from INTEREST
                            group by fieldnum);
```

Question 2. The Relational Model (8 points).

(2 points each)

2.1 Give all likely FDs.
2.2 Give {empID, projID}+ and {deptID}+ based on the FDs in Question 1.
2.3 Specify the primary key (underline) and any foreign keys (*) for each relation.
2.4 Discuss the normal form for the Evaluation relation using FDs in Question 1.

2.1)
deptID -> deptName
deptID -> manager
empID -> empName
empID -> email
empID -> deptID
projID -> deptID
projID -> startYear
empID, projID -> role
~~projID, manager, evalDate ->grade~~, which is reduced to
projID, evalDate --> grade (since projID->deptID, and deptID->manager)

Marking:

- -0.5 for each redundant or missing FD.

2.2)
The discloure should be based on the FDs given in Q2.1.
{empID, projID}+ = {empID,projID, empName, email, deptID, deptName, manager, role, startYear}
{deptID}+ = {deptID, deptName, manager}

Marking:

- 1 point each

2.3)
For each relation:

Department(deptID, deptName, manager*)
Employee (empID, empName, deptID*, email)
Project(projID, startYear, deptID*)
EmpProj(empID*, projID*, role)
Evaluation(projID*, manager*, evalDate, grade)

(Note that manager as a FK references empID.)
(Note that in Evaluation, manager should NOT be part of the PK, since projID -> manager.)

Marking:

- 0.4 for each relation (0.2 for correct PKs, 0.2 for correct FKs)

2.4)
The Evaluation relation is NOT in BCNF or 3NF.
Consider the FD ProjID --> manager (as noted in 2.3)
ProjectID is not a key for the relation (not BCNF) nor is manager (not 3NF)

Marking:

- Need to reference the BCNF and 3NF tests
- Marking based on the FDs in 2.1. (No double penalty (e.g. if manager was wrongly included as the PK for Evaluation in 2.3, and the FD was not reduced in 2.1, then in 2.4 you would conclude that Evaluation IS in BCNF).

Question 3. Normalisation (8 points).

Answer the following questions.

1. (2 marks) Give the minimal basis for the given FDs.
2. (2 marks) Discuss all candidate keys for the APP relation. Explain your answer using the functional dependencies in Question 1.
3. (4 marks) The APP relation is not in BCNF or 3NF. Decompose the relation into BCNF/3NF relations. Your decomposition must keep all functional dependencies and must be lossless. For each resultant relation, discuss if it is in BCNF or 3NF and indicate the primary key (underline) and any foreign keys (*).

Answers:

1. The minimal basis:

docID → docName
patID → patName
patID → patDOB
~~patID, appDate, appTime → docID~~
~~patID, appDate, appTime → roomNo~~
appDate, docID → roomNo
patID, appDate → appTime
~~patID, appDate → roomNo~~
patID, appDate → docID

Note: patID, appDate -> roomNo is redundant because:
Start with: patID, appDate -> docID (last FD)
Augmenting with appDate gives: patID, appDate -> docID, appDate
So: patID, appDate -> docID, appDate
and appDate, docID -> room No (4th FD)
Can transitively create: patID, appDate -> room No
So: patID, appDate -> room No is redundant

Marking:

- -0.5 per error or omission, with a floor of 0

2. The candidate keys: {patID, appDate}. (Marking: 2 marks)

3.Decomposition:
Doctor(<u>docID</u>, docName) BCNF
Patient(<u>patID</u>, patName, patDOB) BCNF
DocRoom(<u>appDate, docID*</u>, roomNo) BCNF
Appointment(<u>patID*, appDate*</u>, appTime, docID*) BCNF

Note: appDate must be PK and FK in one each of DocRoom and Appointment (either way around is fine)

Marking:

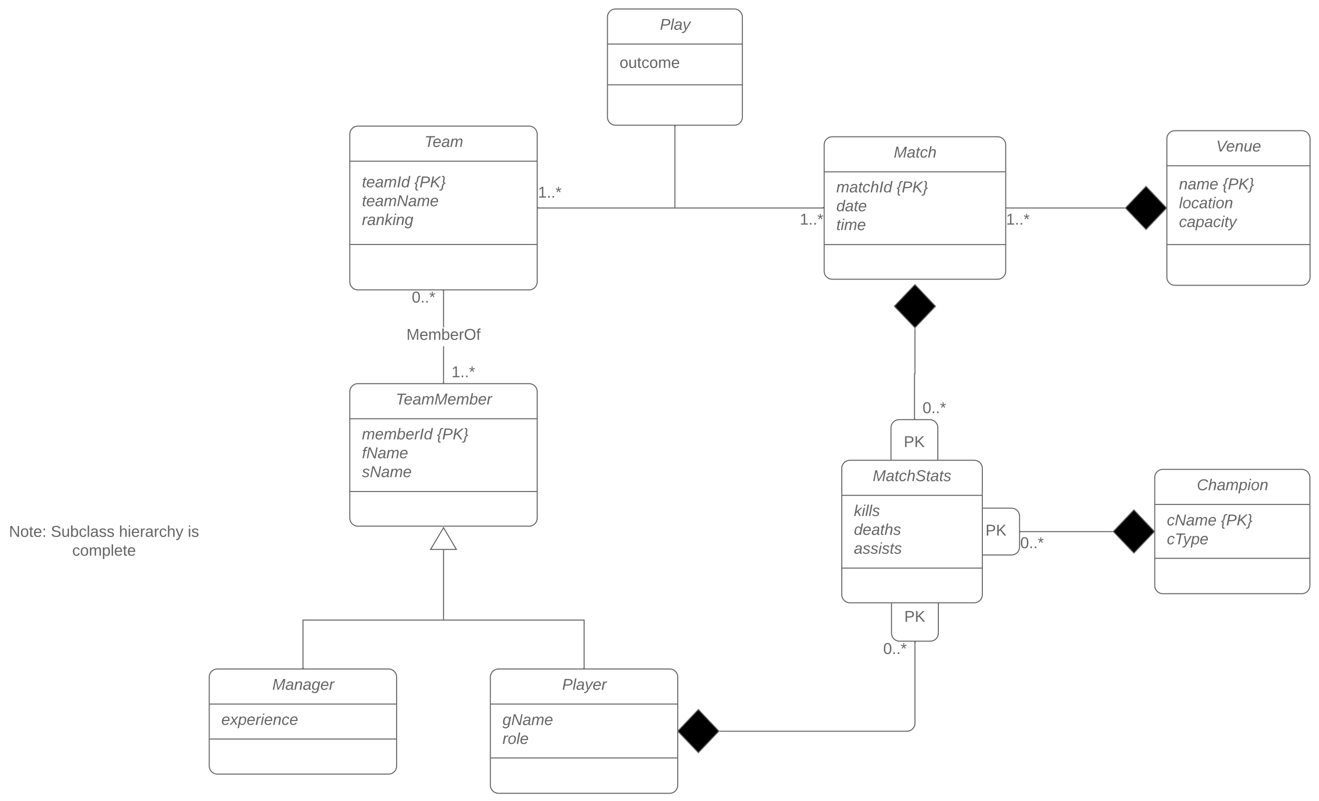
- 1 mark per relation
    - 0.25 for correct attributes
    - 0.25 for BCNF statement
    - 0.25 for correct PKs
    - 0.25 for correct FKs

Question 4. ER Model. (10 points)

- Registered teams compete in the OC. Each team has a name, and a number of team members. A team also maintains a rank throughout the OC, reflecting how well it is doing in the championship.
- Team members are identified by a unique member id, their first name, and their surname. There are two types of team members: managers, and players. For each manager, their number of years of experience is tracked. Each player in a team has an in-game player name, and an assigned role in the team. A role can be one of "top", "mid", "jungle", "adc" or "support".
- A match is a contest between two teams. Teams get a final result, “win” or “lose”, for each match that they play in.
- Matches occur at a specified date and time. They also take place in uniquely named venues. In addition to names, venues have a location and a capacity. Only venues that are part of the OC need to be tracked.
- In a match, each player selects a single champion from a roster of pre-defined champions to play. Each champion may only be selected once in a match. A champion has a unique champion name, and also has a type (for example, “mage” or “support”).
- Additional performance statistics need to be recorded for each champion while being used by a particular player in a particular match. The three statistics to be recorded are a count of the number of "kills", "deaths" and "assists" that they achieve (for example, in a particular match, a particular player playing a particular champion might achieve 4 kills, 2 deaths, and 3 assists).

According to the requirements above, give the ER diagram for the database using the UML class symbols (as used in the lecture notes and tutorials), making appropriate assumptions where necessary. You must represent entities, relationships and their attributes, and all applicable constraints in your diagram. Explain any constraints that cannot be expressed in the diagram.

Oceanic Championship



Marking:

- 2.0 **Entities:** 4 x “Normal” entities with correct attributes – *Team, Match, Venue, Champion* (0.5 each, -0.25 if entity is there but attributes are not correct)
    - *Match* entity:
        - diagram assumes creation of *matchID* attribute as PK for Match, could use date+time+Venue.name as PK instead
        - could optionally include a *numberOfTeamMembers* attribute
    - *Team* entity:
        - diagram assumes creation of *teamID* attribute as PK for Team, could use teamName as PK instead
- 1.5 **Subclass hierarchy:** *TeamMember, Manager, Player* modelled as (0.5 each)
- 0.5 **Note** to state that *subclass hierarchy is complete* (0.5)
- 1.0 *MatchStats* modelled as **weak entity** to enable **ternary relationship** with *PK boxes*
- 1.5 **Relationships** – *MemberOf; Play*; 4 x *Composition* (solid diamond and no name). (0.25 each)
- 0.5 **Association class** (attribute on *Play* relationship) is included
- 3.0 12 **Cardinalities** (0.25 each, at each end of 6 relationships) [*..* also acceptable to indicate many:many where minimum includes 0]
    - *Play* multiplicity on *Team* side: 2..2 or 1..* are fine (representing a specific min/max, or that a match must have a participating team)
    - *MemberOf* multiplicity
        - *TeamMember* side: 1..* or 0..* are fine (representing that a team must have a member to register, or that a team could be registered with no current members)
        - *Team* side: 1..* or 0..* are fine (representing that a player or a manager could be in the database while not currently part of a team)
    - For the *composition* relationships: if marks were already lost for not using composition diamond above, then writing 1..1 is okay here; but should not have both diamond and 1..1 annotation

Question 5. ER to Relational Schema Mapping (9 points).

Course(cno, title)
Student (sno, givename, surname, DOB, address)
Staff (eno, givename, surname)
Tutor(eno, givename, surname, contract)
Enrol (sno*, cno*, grade)
Class(cno*, groupNo, day, time, roomNo, type)
Staff-Class(cno*, groupNo*, staff_eno*)
Tutor-Class(cno*, groupNo*, tutor_eno*)
Take(sno*, cno*, groupNo*)

Notes:

- The Staff-Class and Tutor-Class relations could be combined as one relation:
  - Staff-Tutor-Class(cno*, groupNo*, staff_eno*, tutor_eno*)
- Instead of Staff-Class and Tutor-Class, staff_eno and tutor_eno could also be added as FKs in the Class relation:
  - Class(groupNo, cno*, day, time, roomNo, type, staff.eno*, tutor.eno*)
- The Staff-Tutor subclass relationship is partial. The inheritance by “Tutor” of attributes of Staff and “Staff-Class” relationship are mapped explicitly. Data for Tutor entities and otherwise is kept in separate tables.

Marking:

- If solution includes 9 relations as above: each relation is worth 1 point
- Else if solution includes combined Staff-tutor-class relation (first notes bullet): combined Staff-tutor-class is worth 2 points, others worth 1 point
- Else if solution includes enhanced Class relation with two FKs (second notes bullet): Class is worth 3 points, others worth 1 point

- Any missing attribute, PK or FK in a relation: –0.5 points, with a floor of 0 marks for that relation