COMP 231 - DATABASE MANAGEMENT SYSTEMS

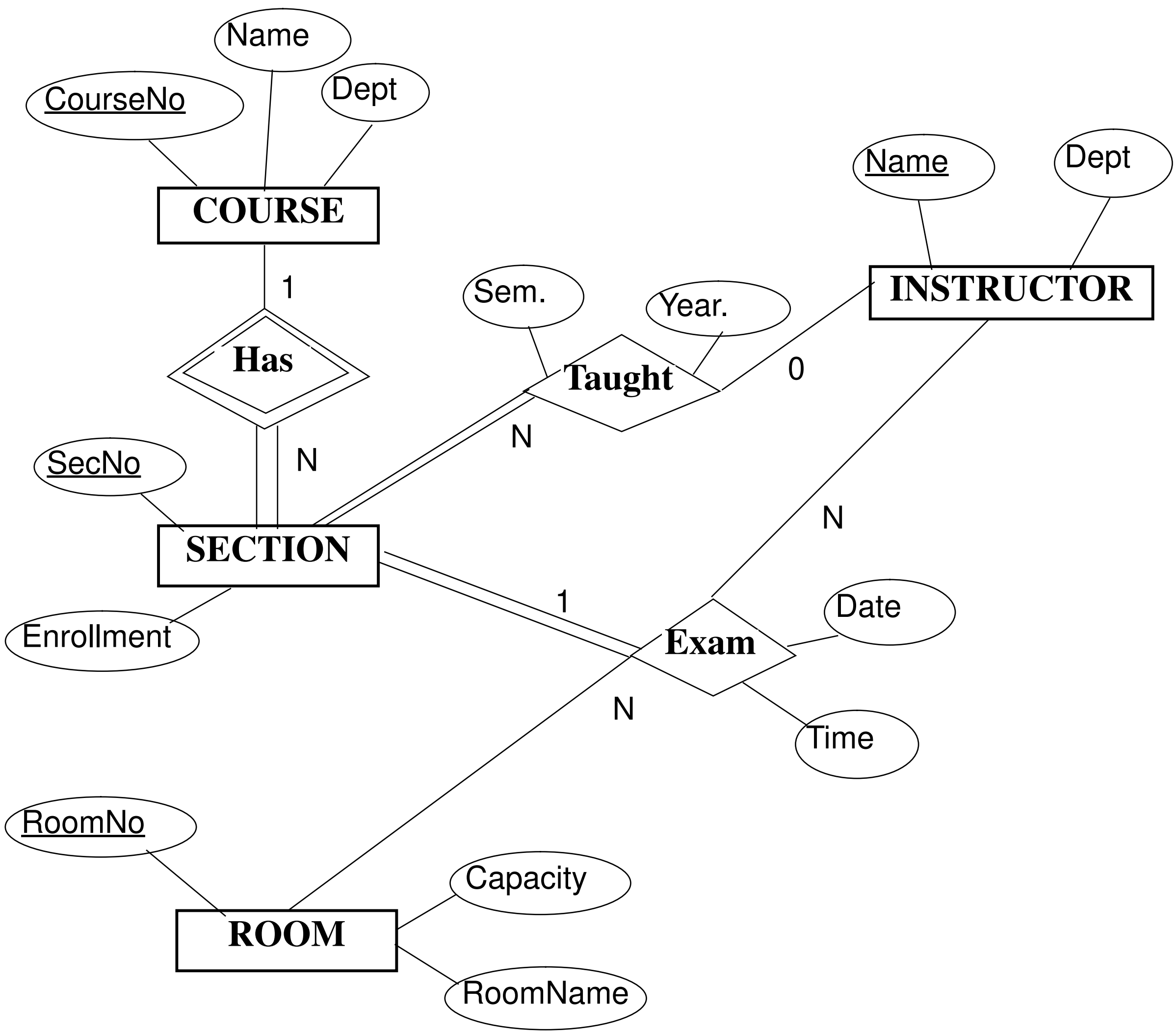
MIDTERM EXAM SOLUTIONS

Note that there can be multiple solutions for the questions, one solution is concisely presented.

I. Entity Relationship Diagram (Sub-Total 25 Points)

Consider a university database for the scheduling of classrooms for final exams.
Course has a unique course number, name, and department
Section of a course has a section number and enrollment. Each course has at least one section.
Room has a unique room number, capacity and a room name.
The name of the instructor and the department he/she belongs to are stored in the database.
A room is allocated for the exam on a section of a course on a given date and time, and the exam is also invigilated by an instructor who teaches the section of a course during a semester of a year.

a) Draw an ER diagram to show the schema for the above set of data requirements. Explicitly show the cardinality and participation constraints and the key attributes. (15 Points)



b) Translate the above ER diagram into a set of relations which are in Third Normal Form. (10 Points)

Step1: Translate each Entity Type to a Relation.

COURSE

CourseNo	Name	Dept

ROOM

RoomNo	RoomName	Capacity

INSTRUCTOR

InsName	Dept

Step 2: Translate each Weak Entity Type to a Relation

SECTION

SectionNo	CourseNo	Enrollment

Step 3: Translate binary relationship types to Relations

TAUGHT

SectionNo	CourseNo	InsName	Semester	Year

Step 4: Translate n-ary relationship type to a relation

EXAM

SectionNo	CourseNo	InsName	RoomNo	Date	Time

Third Normal Form:

List the FDs for each relation above:

CourseNo -> Name, dept
RoomNo -> RoomName, Capacity
InsName -> Dept
SectionNo, CourseNo -> Enrollment
SectionNo, CourseNo, InsName -> Semester, Year
SectionNo, CourseNo, InsName, RoomNo -> Date, Time

There are no more FDs, all these FDs satisfy 3NF condition, therefore, the above relations are in 3NF.

II. Relational Algebra and Query Languages (Sub-Total 55 Points)

Consider the following relational database schema

S (S#, SNAME, STATUS, CITY) -- Supplier relation
P (P#, PNAME, COLOR, WEIGHT, CITY) -- Part relation
SP (S#, P#, QTY) -- Supplier provides Part relation

Write SQL SELECT QUERIES for the following data retrieval statements

a) Get supplier names for suppliers who supply parts with weight greater than 100 grams. (5 Points)

This requires join of all three tables, hence two join conditions, and for parts with weight greater than 100 gms, you have condition Weight > 100.

```
SELECT SNAME
FROM S, P, SP
WHERE S.S# = SP.S# AND P.P# = SP.S# AND WEIGHT > 100
```

b) Get the part names which are not supplied by any of the suppliers (5 Points)

The P# in SP gives all the parts supplied by some supplier, hence to get part names not supplied by any supplier, we have to get P# from P relation which are not in SP relation.

```
SELECT PNAME
FROM P
WHERE P# NOT IN (SELECT P#
                 FROM SP)
```

c) Get the names of the suppliers who supply all red colored parts or supply at least five parts. (10 Points)

Since we need suppliers who supply all red colored parts we have to find all red colored parts and suppliers who supply all of these parts. To get Supplier who supply at least 5 parts we have to count number of parts supplied by each supplier, and this count has to be greater than 4, this is the second part of the query after the UNION.

```
SELECT SNAME
FROM S
WHERE NOT EXISTS (SELECT P#
                  FROM P
                  WHERE NOT EXISTS (SELECT *
                                    FROM SP
                                    WHERE S.S# = SP.S# AND
                                    P.P# = SP.S# AND COLOR = Red))
UNION
SELECT SNAME
FROM S
WHERE S# IN (SELECT S#
             FROM S, SP
             WHERE S.S# = SP.S#
             GROUP BY S#
             HAVING COUNT(*) > 4)
```

Write RELATIONAL ALGEBRA AND TUPLE RELATIONAL CALCULUS EXPRESSIONS for the following data retrieval statements

d) Get supplier name for all suppliers who supply red colored parts (10 Points)

Relational Algebra: JN stands for equi-join, this query requires join of all three relations, then selecting tuples which satisfy part color as red, and then getting the supplier name.
$\pi_{SNAME} (\sigma_{COLOR = RED} (S\ JN_{S.S\#=SP.S\#}\ SP\ JN_{SP.P\#=P.P\#}\ P))$

Tuple Relational Calculus:
{s.SNAME | S(s) AND ∃p ∃t (P(p) AND SP(t) AND s.S# = t.S# AND t.P# = p.P# AND COLOR = RED)}

Get the supplier names who supply all the parts (15 Points)
Relational Algebra: we need to first find all parts, and then apply division operation to find suppliers who supply all parts.
ALL_PARTS = $\pi_{P\#}$ (P)
SUPPLIES_ALL_PARTS = ($\pi_{P\#,\ S\#}$ (SP)) / ALL_PARTS
RESULT=π_{SNAME} (S $JN_{S.S\#=SUPPLIERS\_ALL\_PARTS.S\#}$ SUPPLIES_ALL_PARTS)

Tuple Relational Calculus: We use Universal quantifier, therefore, we ned to filter out unwanted tuples.
{s.SNAME | S(s) AND (∀(p) (not (P(p)) or ((∃t) (SP(t) AND s.S# =t.S# AND t.P#= p.P#))))}

Write a QBE query for the following data retrieval statement

e) Get the names of the suppliers who supply red parts or blue parts which weight greater than 100 grams (10 Point)

S			
S#	SNAME	STATUS	CITY
_SX	P.		

SP		
S#	P#	QTY
_SX	_PX	

P				
P#	PNAME	COLOR	WEIGHT	CITY
_PX		RED	>100	
_PX		BLUE	>100	

_SX and _PX are join variables, the condition required is specified as: conditions on same column are ANDed and conditions on different columns are ORed. Since we want SNAME, P. is in the relation S under column SNAME.

III. Normalization (Sub-Total 35 Points)

Consider the following relation with attributes

Music_CD(CD_Name, Music_Type, CD_Artiste, CD_Track_Number, CD_Song_Writer, CD_Song_Title, CD_Maker, CD_Year, CD_Price)

with following functional dependencies

CD_Name -> CD_Maker, CD_Year, CD_Price
Music_Type, CD_Artiste -> CD_Name
CD_Song_Title -> CD_Song_Writer
CD_Name, CD_Track_Number -> CD_Song_Title, CD_Song_Writer, Music_type, CD_Artiste
CD_Maker, Music_Type -> CD_Price

a) Identify the keys for this relation. Explain why they are the keys. (5 Points)

Let the attributes be renamed from A to I, from left to right. The FDs can be restated as:

fd1: A -> GHI
fd2: BC -> A
fd3: F -> E
fd4: AD -> EFBC
fd5: GB -> I

Note that A^+ = {A,G,H,I}; since AD -> EFBC, we have AD^+ = {A,D,G,H,I,E,F,B,C} (i.e., all attributes of the relation, therefore AD is a key)

Further, since BC -> A; BC^+ = {B, C, A, G, H, I}; since AD -> EFBC, we have BCD^+ = {A,D,G,H,I,E,F,B,C} (i.e., all attributes of the relation, therefore BCD is another key)

The keys are AD, and BCD.

b) What is the normal form of this relation? Explain why is the relation in this normal form? (5 Points)

The relation is in 1NF because all attributes are simple single value attributes. Also, since E, F, G, H, I (non-prime attributes) are not fully functionally dependent (see, FDs fd1, fd4, and fd5) on all the candidate keys the relation is not in second normal form.

c) Normalize this relation to BCNF by first identifying the functional dependencies that violate the normal form (i.e. the functional dependency that makes a relation to be not in 2NF, 3NF, or BCNF), and then normalizing the relation by decomposing the relation.(10 Points)

Since fd1, fd4, and fd5 violate 2NF; we have first decomposition (note that I split fd4 as AD ->EF and AD->BC, to give following decomposition, C2 with two keys & C'2 with one key):

C1(A, G, H, I), C2(A, D(*D*), *B, C*), C'2(A, D, E, F) and C3 (G, B, I), for C2 there are two keys, AD & BCD (D (in plain & in italics), is the same attribute D in two keys AD and BCD respectively). Note in C'2, both E and F are fully functionally dependent on the key AD. The relations C1 & C3 are in BCNF. The FD fd3 violates 3NF condition in C'2, E is transitively functionally dependent on AD. Therefore, further decomposition gives:

C'21(A, D, F) and C'22 (F, E)

The relations C'21 and C'22 are in 3NF and BCNF, but relation C21 is not in BCNF because in

FD fd2: BC -> A, and BC is not a superkey.
Therefore, C21 is further decomposed to:
C211 (<u>D</u>, B, C) and C212 (<u>B, C</u>, A)
Therefore, final BCNF decomposition is:
C1(<u>A</u>, G, H, I); A is key.
C'21(<u>A, D</u>, F); AD is the key. A is foreign key wrt (with respect to) C1. F is foreign key wrt C'22
C'22 (<u>F</u>, E); F is the key.
C211 (<u>D</u>, B, C); D is the key, BC is the foreign key wrt C212
C212 (<u>B, C</u>, A); BC is the key, A is foreign key wrt C1
C3 (<u>G, B</u>, I); GB is the key.

d) Test whether your decomposition has loss less join property. Identify the functional dependencies which are not preserved (if any) in your decomposition (10 Points)

	A (a1)	B (a2)	C(a3)	D (a4)	E (a5)	F (a6)	G (a7)	H (a8)	I (a9)
C1	a1						a7	a8	a9
C'21	a1	*a2 4:(fd4)*	*a3 4:(fd4)*	a4	*a5 3:(fd3)*	a6	*a7 1:(fd1)*	*a8 1:(fd1)*	*a9 1:(fd1)*
C'22					a5	a6			
C211	*a1 2:(fd2)*	a2	a3	a4					
C212	a1	a2	a3				*a7 1:(fd1)*	*a8 1:(fd1)*	*a9 1:(fd1)*
C3	a1						a7	*a8 1:(fd1)*	a9

One column each for an attribute of the relation, one row each for relation in decomposition. The plain values in the table are initial values. The changes are written as follows: in row 2, column 6; the term *a5 3:(fd3)* is interpreted as it is the *3rd* change to the table, it is caused by FD *fd3*, and new value is *a5*.
After four changes to the table, row 2 has all *a* values, therefore, loss-less join property is preserved. The only FD that is not preserved is AD->BC

e) List the set of key constraints, entity integrity constraints, and referential integrity constraints which must be valid on your decomposed relational database schema (5 Points)

Each of the relations in the decomposition has a key. Entity integrity constraints says that key values cannot be null. And for referential integrity constraints all non-null foreign key (see above (end of solution for part (c)) for foreign keys for decomposed relations) values must exist in the parent relation.