Database Systems (CS F212)
MidSem Solutions

Q1. (a) If all sets of attributes are closed, then there cannot be any nontrivial functional dependencies. For suppose A1A2...An->B is a nontrivial dependency. Then A1A2...An+ contains B and thus A1A2...An is not closed.

Non trivial FDs – **3 marks**
Contradiction – **2 marks**

(b) Let R(A,B) be a 2-attribute relation. The possibilities for keys are A, B, both A & B are keys separately, & composite AB. – **1 mark**
If A (or B) is the only key (therefore PK), then the only FD is A→B (or B→A). So BCNF. – **1 +1 mark**
If AB is the composite key, then the only trivial FD is AB→AB. So BCNF. – **1 mark**
If A and B, both are CKs, then the FDs are A→B, and B→A. Determinants in both FDs are CKs. So BCNF. – **1 mark**

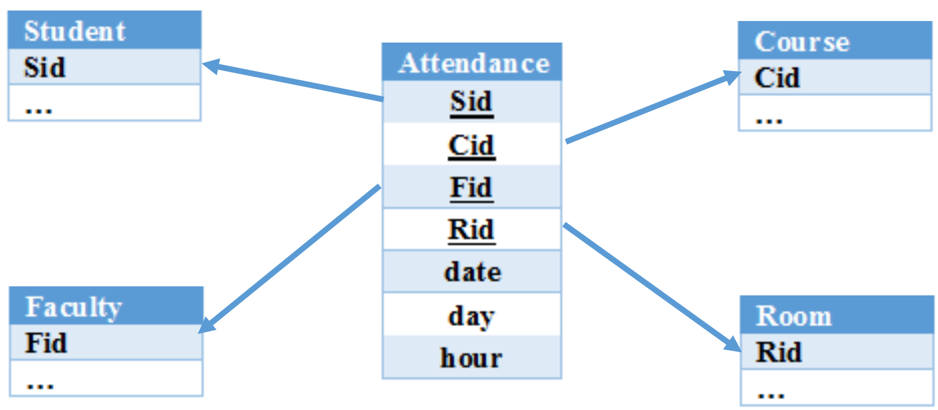
Q2

a) Attendance (sid, cid, fid, rid, date, day, hour) (2 Marks, If PK not specified deduct 1 Marks)(If finest granularity not considered, 1 marks deducted)
student(Sid, name,…) (1/2 Marks, If PK not specified 0 Marks)
course(cid, name,…) (1/2 Marks, If PK not specified 0 Marks)
faculty(fid, name,…) (1/2 Marks, If PK not specified 0 Marks)
room(rid,…) (1/2 Marks, If PK not specified 0 Marks)

b) Functional Dependencies
PK Dependency (2 Marks)
(If a) is not correct/ there is a problem with functional dependencies 1Marks)

c) Normalize it to BCNF
Already in BCNF (2 Marks) (If a) is not correct/ there is a problem with normalization 1 Marks)

d) Referential Integrity Diagram



(2 Marks)
(1/2 Marks each of the table, with PK mentioned)

Q6

(a) FD$_s$ for the relation:
Primary Key Dependency
sub_category -> category **(1 mark)**
category -> department **(1 mark)**
(If PK not written ½ marks deducted)

(b) Identifying all redundancies:
Category and Department information getting repeated for each product which should not be the case.
with reason for both is given (1 mark + 1 mark)

(c) Normalized relation:
Table1 : prod_key, prod_name, prod_decription, sub_category
Table2: sub_category, category
Table3: category, department
(For All three tables 2 Marks, For 2 correct tables 1 marks)
Tables are normalized to BCNF Form (1 Marks)

(d) Space Saving:
Before Normalization: 60,000 × 25 × 6 = 9,000,000 bytes (1 marks)
After Normalization: (60,000 × 25 × 4) + (600 × 25 × 2) + (60 × 25× 2) = 6033000 bytes (1 marks)
Space Saving : 2967000 bytes (1 Marks)

Q3. DDLs to create/specify FDs. For Example:

Create FDs - 2 marks

R.A ➔ R.BC as R.FD.1 (name optional) – 1 mark

R.C ➔ R.D as. R.FD.2

R.C ➔ R.E as R.FD.3

(can be done at the time of creation of R or any time later using Alter Table) – 1 mark

Any FD violations are flagged and corresponding update operation is denied.

Every FD on every relation is stored in the metadata, like any other DB Object. – 2 marks

Normalize R into R1, R2 using R.C ➔ R.D (or using name R.FD.1) – 2 marks

Dropping a FD – 1 mark

Drop R.FD.1 - 1 mark

Q4

Given :

Block Size (B) = 512 Bytes

Block Pointer (P) = 6 Bytes

Record Pointer (P_R) =7 Bytes

Record Size (R) = 114 Bytes

No. of Records (r) = 30,000

(a) Taking 'SSN' as the key field

(i) Index Blocking factor:

For an index on the SSN field, field size V_{SSN} = 9 bytes, block pointer size P =6 bytes.

Then: index entry size R_i=(V_{SSN}+ P)=(9+6)=15 bytes

Index blocking factor Bfr_i= B/R_i= 512/15 = **34 entries/block**

--1/2M

(ii) Number of First Level Index Entries:

Bfr = 512/114 = 4 records/block

number of file blocks b= (r/Bfr)= (30000/4) = 7500

Therefore, r_1 = **7500 entries**

--1/2 M

Number of First level index Blocks:

number of index blocks b_1= (r_1/ Bfr_i)= (7500/34) = **221 blocks**

--1/2M

(iii) Number of levels needed if we make it into a multi-level index:

Number of First level index Blocks:

number of index blocks b_1= (r_1/ Bfr_i)= (7500/34)= 221 blocks

Number of Second level index Blocks:

number of index blocks b_1= (r_2/ Bfr_i)= (221/34)= 7 blocks

Number of Third level index Blocks:
number of index blocks b_1= (r_3/ Bfr_i)= (7/34)= 1 block
Therefore, 3 levels
--**2 M**

(iv) Number of Blocks required by the multi-level index:
number of blocks = 1+7+221 = **229 blocks**
-- 1M

(v) Number of block accesses needed to search for and retrieve a record from the file –given its SSN value—using the primary index:
3 index block accesses + 1 data block = 4
--1/2M

(b) Using secondary index on SSM

(i) Index Blocking factor:
For an index on the SSN field, field size V_{SSN} = 9 bytes, block pointer size P =6 bytes.
Then: index entry size R_i=(V_{SSN}+ P_R)=(9+7)=16bytes
Index blocking factor Bfr_i= B/R_i= 512/16 = **32 entries/block**
--1/2M

(ii) Number of First Level Index Entries:
Therefore, r_1 = **30,000 entries**
Number of First level index Blocks:
number of index blocks b_1= (r_1/ Bfr_i)= (30000/32) = **938 blocks**
--**1/2M**

(iii) Number of levels needed if we make it into a multi-level index:
Number of First level index Blocks:
number of index blocks b_1= (r_1/ Bfr_i)= (30000/32)= 938 blocks
Number of Second level index Blocks:
number of index blocks b_2= (938/(34))= 27 blocks
Number of Third level index Blocks:
number of index blocks b_3= (27/(34))= 1 block

Therefore, 3 levels
--**2 M**

(iv) Number of Blocks required by the multi-level index:
number of blocks = 1+27+938 = **966 block**
--1/2M

(v) Number of block accesses needed to search for and retrieve a record from the file –given its SSN value—using the primary index:
3 index block accesses + 1 data block = 4
--1/2M

Comparison:

	Primary index is better than secondary index --------------------------1/2M **Because: space requirement by PI is much less than SI, although the no. of block acceses needed to search for a SSN value by both is same. 1/2M**
Q5	Given: R(A, B, C, D) FD_s (A->AC, B->ABC, D->ABC) Steps to find F_c : 1. Simpleton RHS {A->A, A->C, B->A, B->B, B->C, D->A, D->B, D->C} Therefore, F = { A->C, B->A, B->C, D->A, D->B, D->C} F = { A->C, B->AC, D->ABC} **-- 2M** 2. Removal of Extraneous attributes Checking for A in B->AC (A is not extraneous) Checking for C in B->AC (C is extraneous) **--3M** Checking for A in D->ABC (A is extraneous) Checking for B in D->ABC (B is not extraneous) Checking for B in D->ABC (A is extraneous) **--3M** 3. Removal of redundant FD_s Therefore, **F_c (A->C, B->A, D->B)** **--2M**