Section 1: Redundancy Anomalies [13 points]

1- (9 points) Consider the following table. Give an example of update anomaly, an example of deletion anomaly and an example of insertion anomaly knowing that the *Group* is determined by the *Age*.

ID	Name	Age	Group	Address
123	John	35	A	Edmonton
321	Alfred	45	B	Calgary
192	Hang	20	C	Regina
918	Sophie	45	B	Vancouver
789	Jane	19	C	Edmonton

Update Anomaly: **(3 points)**

Changing the group in record 321 into <321, Alfred, 45, C, Calgary> would generate inconsistency since record 918 would also have to change.

Deletion Anomaly: **(3 points)**

Removing record 123 would make us loose the information that age 35 determines a group A.

Insertion Anomaly: **(3 points)**

Adding a record when we know the age but not the category is a problem: <999, Tom, 55, ?, Montreal> or simply adding the information that age 100 determines a group Z is impossible.

or

Adding a new group is not possible until we add a record of someone belonging to the group.

2- (4 points) Give a schema of a decomposition that avoids these anomalies.

R1(ID, Name, Age, Address) R2(Age, Group)

Section 2: Scheduling Transactions [17 points]

1- [8 points] Suppose we have two bank accounts A and B and we want to transfer \$20 from A to B. We could do it with a transaction
T1: R(A); A=A-20; R(B); B=B+20; W(A); W(B).
Suppose we also want to add 2% interest to all accounts. This could be done with a transaction T2: R(A); A=A*1.02; R(B); B=B*1.02; W(A); W(B).
Give a serial schedule for these transactions, and an equivalent non-serial schedule for these transactions.

Serial Schedule (4 points)

```
T1:  R(A) R(B)  W(A)  W(B)
T2:                         R(A)  R(B)  W(A)  W(B)
```

Equivalent Non-serial Schedule (4 points)

```
T1:  R(A)  R(B)  W(A)       W(B)
T2:                    R(A)      R(B)  W(A)  W(B)
```

2- [4 point] Why is it important to interleave transactions?

Faster concurrent transactions

3- [5 points] Briefly explain what is Atomicity and enumerate the other remaining ACID properties. You don't have to explain C, I and D.

Atomicity: **All or nothing: All operations of a transactions are executed or none. (2 points)**

C stands for **Consistency (1 point)**

I stands for **Isolation (1 point)**

D stands for **Durability (1 point)**

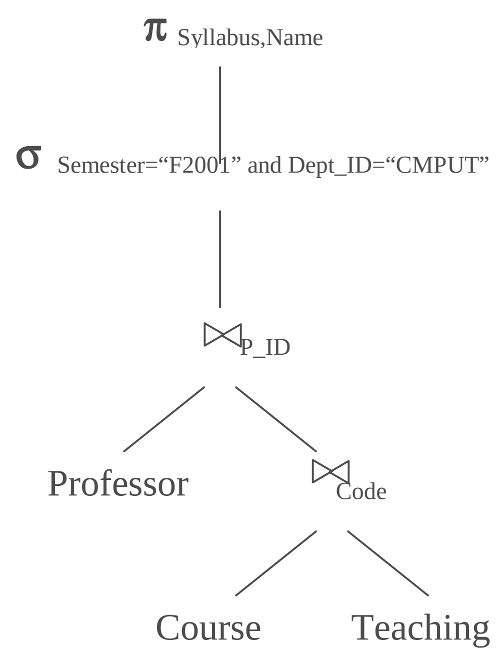
Section 3: Query Optimization [43 points]

Given the following relations for the entities Professor and Course and the relationship Teaching:

Professor (P_ID, Name, Dept_ID)
Course(Code, Dept_ID, CName, syllabus)
Teaching(P_ID, Code, Semester)

1- [6 points] Give the equivalent query tree for this relational algebra expression:

$$\pi_{\text{Syllabus,Name}}\ (\sigma_{\text{Semester="F2001" and Dept\_ID="CMPUT"}}\ (P \bowtie_{\text{P\_ID}} (C \bowtie_{\text{Code}} T)))$$

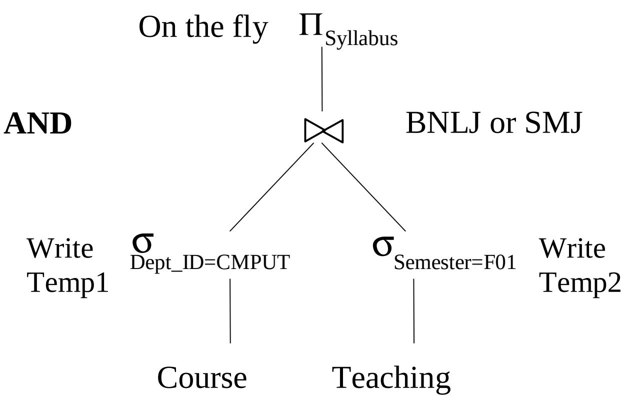


2- [5 points] Give an alternative relational algebra expression equivalent to the previous one in question 1 such that the selection is done as early as possible.

$$\pi_{\text{Syllabus,Name}}\ ((\sigma_{\text{Dept\_ID="CMPUT"}}(P)) \bowtie_{\text{P\_ID}} ((\sigma_{\text{Dept\_ID="CMPUT"}}(C)) \bowtie_{\text{Code}} (\sigma_{\text{Semester="F2001"}}(T))))$$

3- [32 points] Given the SQL query and the unfinished query plan below, estimate the query execution I/O cost using a Bloc-Nested Loop Join (BNLJ) and using the Sort-Merge Join (SMJ).

SELECT C.syllabus
FROM Course C, Teaching T
WHERE C.Code = T.Code **AND**
T.Semester = “Fall 2001” **AND**
C.Dept_ID = “CMPUT”



There are 2500 courses in 100 departments. The relation Course is contained in 500 pages of disk, each page with 5 tuples of Course. There are 3000 teaching records for 3 semesters. The relation Teaching is contained in 300 pages, each page with 10 tuples of Teaching. Suppose we have 4 buffer pages (blocs) in main memory, and assuming a uniform distribution for all the values in the database, estimate the cost in terms of I/O of the execution plan above.

Cost of Selections (Scan + writing Temp files)
Scanning course =500 i/o. Writing Temp1=500/5= 5pages => 505i/o
(4 points)
Scanning Teaching =300 i/o. Writing Temp2=300/3=100pages =>400i/o
(4 points)
Cost of BNL join:
We have 4 blocs => 2 blocs for reading Temp1 => read Temp2 3 times ($\lceil 5/2 \rceil$) and of course read Temp1 once => 305 i/o
(7 points)
Total cost of query plan with BNLJ = 505+400+305 = 1210 i/o
(1 point)

Sorting Temp1: 4 blocs(memory)& 5 disk pages => 2 passes
=> reading & writing all blocs in each pass => 2*2*5 = 20 i/o
(6 points)

Sorting Temp2: 4 blocs (memory)& 100 disk pages => 5 passes
=> reading & writing blocs in each pass => 2*5*100 = 1000 i/o
(6 points)

Merge the sorted Temp1 and Temp2 = 100 + 5 pages = 105 i/o
(3 points)
Total cost of query plan with SMJ = 505+400+20+1000+105= 2030 i/o
(1 point)

Section 4: Functional Dependencies [27 points]

1- [12 points] Consider the following relation R(A,B,C,D,E,F,G,H), the candidate keys AB, CD, E, and the functional dependencies A→ F.

a- Show that the functional dependency violates BCNF using only the definition and properties of BCNF.

(4 points)

```
in FD A→F  A does not contain a key
           or A is not a super-key
```

b- Show that the functional dependency violates 3NF using only the definition and properties of 3NF.

(4 points)

```
in FD A→F  A does not contain a key
           or A is not a super-key

           and F is not a key attribute
```

c- Show that the functional dependency violates 2NF.

(4 points)

```
A is part of a key (AB)
A→F : Part of a key determines a non-key attribute(F)
```

2- [9 points] Based on the relation schema and functional dependencies of the previous question, give a lossless join and dependency preserving decomposition that generates relations in BCNF. Show that it is lossless join decomposition and explain why it is dependency preserving.

R1(A,F) R2(A, B, C, D, E, G, H) **(4 points)**

$R1 \cap R2 = A$ A is key for R1 => lossless join decomposition **(3 points)**

Only one functional dependency $A \rightarrow F$. It is preserved in R1.
(2 points) $(F_{R1} \cup F_{R2})^+ = F_R^+$

3- [6 points] Given a relation with the following schema R(A, B, C, D) and the following functional dependencies, use Armstrong axioms and the other derived rules (union and decomposition) to prove that C is a candidate key. Show every step and indicate the rule used.

$B \rightarrow A$
$C \rightarrow D$
$C \rightarrow B$

$C \rightarrow C$ reflexivity - trivial **(2 points)**

$C \rightarrow B$ and $B \rightarrow A$ => $C \rightarrow A$ transitivity **(2 points)**

$C \rightarrow A$, $C \rightarrow B$, $C \rightarrow C$, and $C \rightarrow D$ => $C \rightarrow ABCD$ Union **(2 points)**