Section 1: Redundancy Anomalies [10 points]

1- (6 points) Consider the following table. Give an example of update anomaly, an example of deletion anomaly and an example of insertion anomaly knowing that a product has many suppliers and can have many other products as a substitute (i.e. a product can be replaced by its substitute). The purchase price is determined by a supplier for a product, while the sale price is for a given product regardless of the supplier. The quantity is for a given product, again regardless of the supplier.

ProductID	SupplierID	Substitute	Quantity	PurchasePrice	SalePrice
123	987	121	509	200	250
123	987	122	509	200	250
123	998	121	509	210	250
123	998	122	509	210	250
342	987	344	120	99	189
432	789	433	100	110	200

Update Anomaly:

- Changing the quantity of a product implies updating the quantity for as many suppliers and substitutes there is for the product.

Deletion Anomaly:

- By deleting the only substitute of a product, the whole product entry needs to be removed.

Insertion Anomaly:

- We can't add a substitute of a product if we do not know the supplier of the product.

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

Product(ProductID, Quantity, SalePrice)
Suppliers(ProductID, SupplierID, PurchasePrice)
Substitutes(ProductID, Substitute)

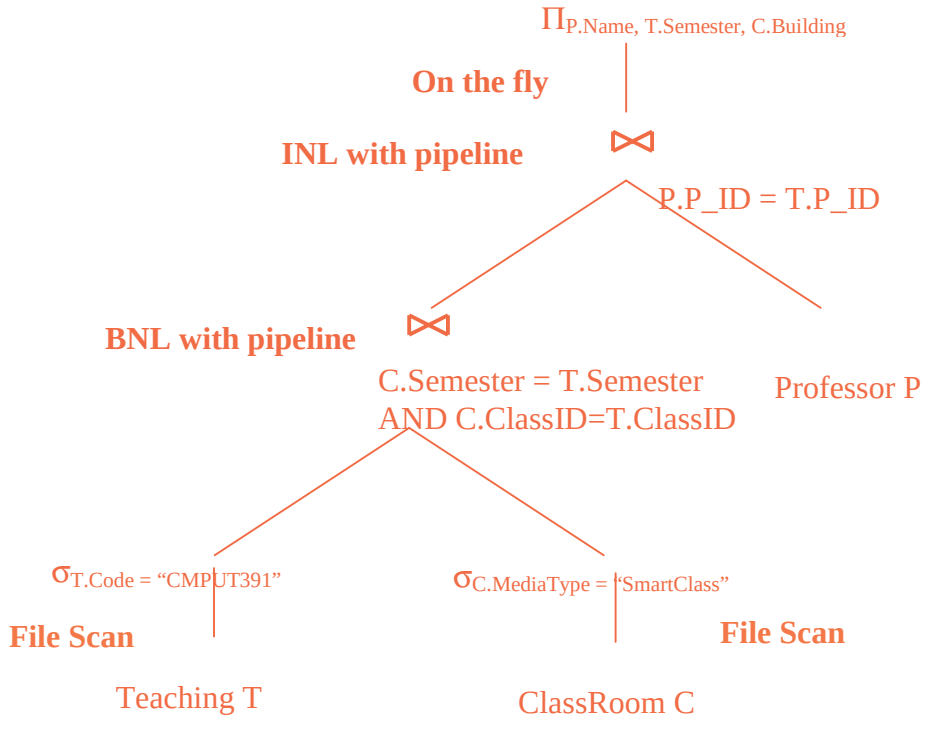
Section 2: Query Optimization [40 points]

Given the following relations Professor, Course, Teaching and ClassRoom:

Professor (P_ID, Name, Dept_ID)
Course(Code, Dept_ID, CName, syllabus)
Teaching(P_ID, Code, Semester, ClassID)
ClassRoom(ClassID, Semester, Building, MediaType)

1- [8 points] Given the following SQL query, draw the query plan based on a left-deep tree where all selections are done as early as possible, a block-nested loop join (BNLJ) for the result of the selections and an index nested loops join strategy afterwards, knowing that the relation Professor is indexed on P_ID. In order to decide which relation is the outer for the BNLJ, consider that there are about 30 semesters in the database and cmput391 is at most taught twice a semester. However, there are several hundreds of smart classrooms (they wouldn't fit in main memory).

SELECT P.Name, T.Semester, C. Building
FROM Professor P, Teaching T, ClassRoom C
WHERE P.P_ID = T.P_ID **AND** C.Semester=T.Semester **AND**
T.ClassID=C.ClassID **AND** C.MediaType="SmartClass" **AND**
T.Code = "CMPUT391"



(5 points for the query tree, 3 points for the plan)

2- [20 points] Suppose we have 10 buffer pages (blocks) in main memory, and assuming a uniform distribution for all the values in the database (except for classroom mediatype, where 60% of classrooms are smart classrooms), estimate the query execution I/O cost for the above plan.

Relations	# of tuples in the relation	# of tuples per page
Professor	200	8
Course	1,000	5
Teaching	25,000	10
ClassRoom	1,006	10

There is no index on attribute Code for the relation Teaching and the relation is neither clustered nor sorted on this attribute. Professor is indexed on P_ID using a clustered hash index. All CMPUT391 teachings are in smart-classrooms.

12 points

8 points

First Join: BNLJ **(20 points)**

Selection needs to read all Teaching: 2500 I/O

The size of the result is 25000/1000 = 25 teaching for CMPUT391 = 3 pages. There is enough memory to keep the 3 pages of CMPUT391 teachings in main memory while selecting the classrooms. — 3.5

Selection needs to read all ClassRoom: 101 I/O — 3.5

Cost for first join after selection = 2500 + 101 = 2601 I/O
Alternatively, if the selection classrooms is materialized, we need to write 61 pages (1006*60%/10) and re-read them once. That is 2500 + 101 + 122 = 2723 I/O — 5

Second Join: INLJ

All cmput391 courses are in smart classrooms. That means we still have 3 pages of teachings. They can fit in main memory and the result is pipelined to the second join. For each teaching record, we need to access the hash for professors. That is 25 teaching. Cost of join is (25,000/1,000)*1.2 = 25 * 1.2 = 30 I/O since the index is clustered.

Finally, the projection is done on the fly. The total cost = 2601 + 30 = 2631 I/O

Alternatively if the selection on classrooms is materialized then the total cost would be 2631 + 122 = 2753 I/O

3- [2 points] What would have been the total cost had we had an un-clustered hash index on P_ID for Professor? Justify your answer.

If the index is not clustered, we need an extra I/O for each professor to access the data. That is an additional 25 pages. The Total cost would be 2601 + 30 + 25 = 2656

4- [10 points] Using the same databse as in questions 1 to 3, consider the following query:

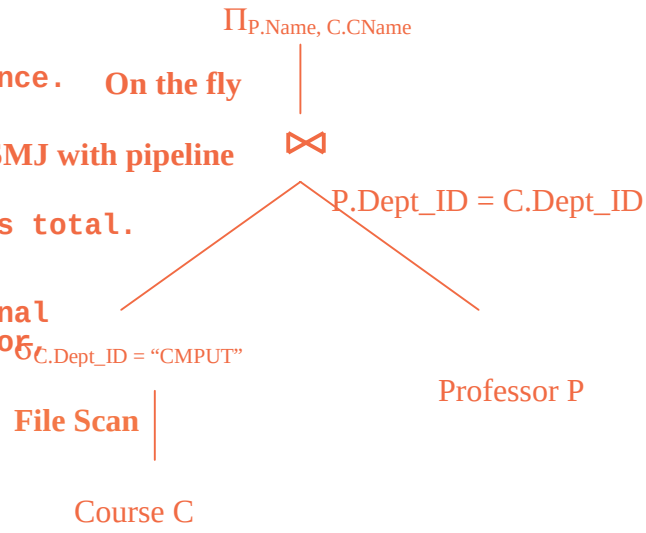
SELECT C.CName, P.Name
FROM Professor P, Course C
WHERE C.Dept_ID = P.Dept_ID **AND**
C.Dept_ID=”CMPUT”

Suppose we have 40 departments. Give the query plan for a sort-merge join and estimate the cost to execute this query as it is specified assuming the selection on department for courses is done early.

We need to sort the professors by Dept_ID.
Professor has (200/8)=25 pages, with 10 blocks in main memory we get 3 runs that we can merge in one iteration that is 4 times 25 = 100 I/O to sort Professor.

Selecting the courses in CS necessitates reading Course once. That is 200 I/O

The result is 1000/40 tuples
With 5 per page that’s 5 pages total.
It can fit in main memory.
The sort-merge join will cost
In the worst case the additional reading of the sorted Professor, that is 25 pages.
Since the projection is on the fly, the total cost is 100 + 200 + 25 = 325 I/O



Sort prof=3

Select CS=1

Plan = 3

Sort-Merge=2

Exact total=1

5- [5 points] Is there a better plan? Justify your answer.

A Block-Nested Loop Join would do the trick. The cost would be scanning Course (200 pages) to select the courses in CS. Since these would fit in main memory (100/40/5 = 5 pages), we can scan Professor once (25 pages) to do the join. The total would be 225 I/O.

Yes
BNLJ=3
Correct justification =2

Section 3: Functional Dependencies [35 points]

1- [16 points] Consider the following relation R(A,B,C,D,E,F,G,H), the candidate keys A, DE, and, DF, and the functional dependency E→ F.

a- [4 points] Does the functional dependency violate BCNF? Justify your answer using only the definition and properties of BCNF.

(4 points)

E→F violates BCNF because
it is not trivial and E does not contain a key

b- [4 points] Does the functional dependency violate 3NF? Justify your answer using only the definition and properties of 3NF.

(4 points)

E→F does not violate 3NF because F is a key attribute

c- [8 points] Assume now, that another functional dependency G->F holds in addition. Show that now DG is also a key for the relation R in the following two ways

- Using Armstrong's Axioms (and possibly union and decomposition rule). In each step of the proof make the used axiom/rule explicit.
- Using the Attribute Closure algorithm. Show the accumulation of the closure in each step of the algorithm.

(4 points)

Trivially DG→DG. By decomposition DG→G. DG→G and G→F imply by transitivity DG→F. By augmentation with D: DG→DF. DF is a key, i.e. DF→R. Hence by transitivity DG→R.

(4 points)

DG {DG}
G→F {DGF}
DF→R {ABCDEFGH}

2- [19 points]

a- [9 points] Based on the relation schema and functional dependencies of question 1a and 1b, 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(E,F) R2(A, B, C, D, E, G, H) **(4 points)**

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

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

b- [3 points] Now assume again, as in question 1c, that the functional dependency G->F holds in addition. Is this dependency preserved by your decomposition. Justify your answer.

G → F. It is NOT preserved.
(3 points) $(F_{R1} = \{E \rightarrow F\} \cup F_{R2} = \{A \rightarrow R_2, DE \rightarrow R_2\})^+ \neq F_R^+$

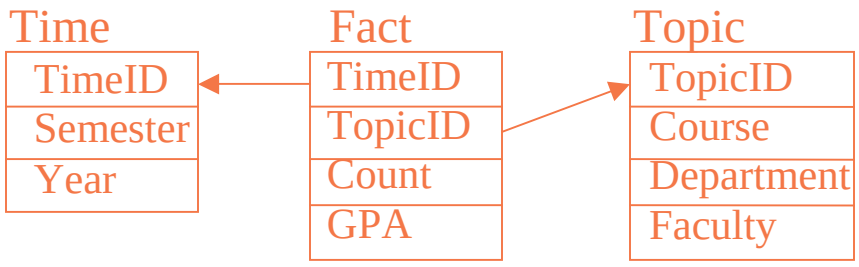
3- [7 points] Prove the following generalized transitivity rule: If Z⊆Y, then (X→Y and Z→W) entail X→W. Prove this rule using the arguments that directly appeals to the definition of FDs,

Let t, s be arbitrary tuples in a relation that satisfy the above conditions and that agree on X. Then, due to X → Y, they agree on Y and thus also on Z. But then, due to Z → W, t and s must agree on W. Therefore, any pair of tuples t and s that agrees on X in such a relation must also agree on W, i.e., X → W must hold.

Section 4: Data Warehousing [10 points]

1- [6 points] Give the schema of the relations that would implement a star-schema for a data warehouse about grades at a university. The measures are the number of students and the average GPA, both for a given time and topic. Time is organized in semesters and years, while topic is organized in course, department, and faculty.

```
Fact(Count,GPA,TimeID,TopicID)
Topic(TopicID, Course, Department, Faculty)
Time(TimeID, Semester, Year)
```



2- [4 points] Give a single SQL:1999 query that would generate a table with the information equivalent to the following table. The data in the cells are counts.

	Arts	Science	Education	Engineering	**Total**
2000	2504	3010	502	603	**6619**
2001	2780	3104	489	615	**6988**
2002	2867	2901	492	584	**6844**
Total	**8151**	**9015**	**1483**	**1802**	**20451**

```
SELECT T.Year, F.Faculty, SUM(F.Count)
FROM Fact F, Time T, Topic P
WHERE F.TimeID=T.TimeID AND F.TopicID=P.TopicID
GROUP BY CUBE (T.Year, F.Faculty)
```