Department of Computer Science
University of KarachiFinalEXAM(BSCS-F) 17th Nov 2023

Course supervisor: Dr. Nadeem Mahmood **Course No & TITLE:BSCS- 606 "Distributed Database System"**
Time: 2 Hours **Total Marks:** 70
Attempt any FOUR Questions: All questions carry equal marks. Switch off your mobile phones.

Question 1

a) Discuss Distributed database reference architecture and also elaborate autonomy and heterogeneity w.r.t the model
b) Given a relation R(K;A;B;C) (where K is the key) and the following query

SELECT *FROM RWHERE R.A = 10 AND R.B=15

i. What will be the outcome of running PHF on this query?
ii. Does the COM MIN algorithm produce in this case a complete and minimal predicate set? Justify your answer.

Question 2

a) Describe how the following can be properly modeled in thedatabase allocation problem.
i. Relationships among fragments
ii. Query processing
iii. Integrity enforcement
iv. Concurrency control mechanisms

Simplify the following query:and transform it into an optimized operator tree using the restructuring algorithm where select and project operations are applied as soon as possible to reduce the size of intermediate relations.

```
SELECT      ENAME, PNAME      FROM      EMP, ASG, PROJ
WHERE       (DUR > 12 OR RESP = "Analyst")   AND      EMP.ENO = ASG.ENO
AND         (TITLE = "Elect. Eng."      OR      ASG.PNO < "P3")
AND         (DUR > 12 OR RESP NOT= "Analyst")AND      ASG.PNO = PROJ.PNO
```

Question 3

a) Some architectural models favor the definition of a global conceptualschema, whereas others do not. What do you think? Justify your selection withdetailed technical arguments.

b) Give the query graph of the following query, in SQL, on our exampledatabase:
SELECT ENAME, PNAME FROM EMP, ASG, PROJ WHERE DUR > 12 AND EMP.ENO = ASG.ENO AND PROJ.PNO = ASG.PNO
and map it into an operator tree.

Question 4

a. Differentiate:
i. LAV vs GAV ii. Schema vs instance matching iii. Join vs semi joins

b. Consider three sources:

Database 1 has one relation Area(Id, Field) providing areas of specialization ofemployees; the Id field identifies an employee.
Database 2 has two relations, Teach(Professor, Course) and In(Course, Field);Teach indicates the courses that each professor teaches and In that specifiespossible fields that a course can belong to.
Database 3 has two relations, Grant(Researcher, GrantNo) for grants given toresearchers, and For(GrantNo, Field) indicating which fields the grants are for.
The objective is to build a GCS with two relations: Works(Id, Project) statingthat an employee works for a particular project, and Area(Project, Field) associatingprojects with one or more fields.
(a) Provide a LAV mapping between Database 1 and the GCS.
(b) Provide a GLAV mapping between the GCS and the local schemas.
(c) Suppose one extra relation, Funds(GrantNo, Project), is added to Database 3.Provide a GAV mapping in this case.

Question 5

a. Consider the join graph of figure A and the following information:size(EMP) = 100, size(ASG) = 200, size(PROJ) = 300, size(EMP 1 ASG) = 300,and size(ASG 1 PROJ) = 200. Describe an optimal join program based on theobjective function of total transmission time.
b. Consider the join graph of the following figure A, and give a program (possiblynot optimal) that reduces each relation fully by semijoins

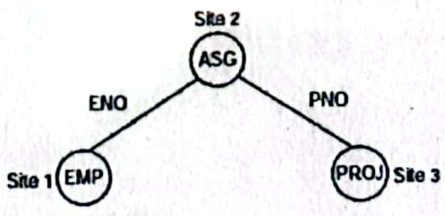


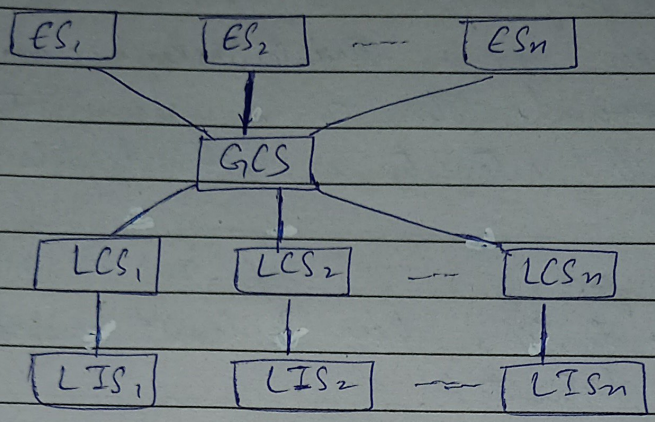
Figure A: Join Graph of Distributed Query

P.P(2023) Q1.a

Date

Q) Discuss distributed database reference architecture & also elaborate autonomy & heterogeneity w.r.t the model.

=> Distributed Database Reference Architecture :-



→ ES (External Schema) : The user's view of the database, shows data in a way that each user or application needs.

→ GCS (Global Conceptual Schema) : Provide a unified view of all distributed data across sites.

→ LCS (local Conceptual Schema) : Discribes how data is stored locally structured in each database.
individual site.

→ LIS (local Internal Schema) : Defines. The actual physical storage structure in each local Database

=> Autonomy : defines how independent each site is.
- Each site's DBMS (LCS/LIS) controls its own data & operations
- GCS coordinates across sites but does not fully control local operations.

=> Heterogeneity : refers to differences among sites in DBMS type, data model, or platforms.
- The architecture handles this via the global schema (GCS/ES) & communication layers, allowing integration despite local differ

→ P.P(2023) Q1 .b

3.10
```
SELECT *
FROM R
WHERE R.A=10 AND R.B=15
```

(a) What will be the outcome of running PHF on this query?

PHF: $p_1 : A=10$, $p_2 : B=15$

minterms

$m_1 : p_1 \wedge p_2$ $m_2 : \neg p_1 \wedge p_2$ $m_3 : p_1 \wedge \neg p_2$ $m_4 : \neg p_1 \wedge \neg p_2$

→ Given query will access only $m_1 : p_1 \wedge p_2 \Rightarrow A=10 \wedge B=15$ fragment.

(b) Does COM-MIN Algo produce in this case a complete & minimal predicate set? Justify your answer.

→ Yes, COM-MIN Algo is complete (covers all data in query) & minimal (no unnecessary/extra conditions)

→ COMMIN: return {A=10, B=15} Here, that set is complete & minimal for the access pattern given in the query.

RC

No.

P.P(2023) Q2.a

Date

3.15: Describe how the following can be properly modeled in db allocation problem.

(a) Relationships among fragments :-

When data is fragmented, the fragments are still logically related through common attributes or keys or join constraints, to ensure the original relation can be reconstructed using joins & that referential integrity is preserved b/w related fragements.

(b) Query Processing :-

Query Processing identifies where & how to execute a query when data is distributed.

It includes:

- Data localization : Identify which fragements are needed.
- Query shipping : send query to the site(s) that hold those fragment.
- Result assembly : Combine partial results from different sites

→ Goal : minimize communication cost & respone time.

(c) Integrity enforcement :-

Ensures data correctness across sites. Keeps data consistents & validate across all fragments & replicated copies.

(d) Concurrency Control Mechanisms :-

Handle simultaneous access to data at different sites.
Prevent conflicts when many users update fragments at the same time.

→ Goal : maintain transaction consistency & avoid conflicts across fragments.

→ P.P (2023)

Q2.b

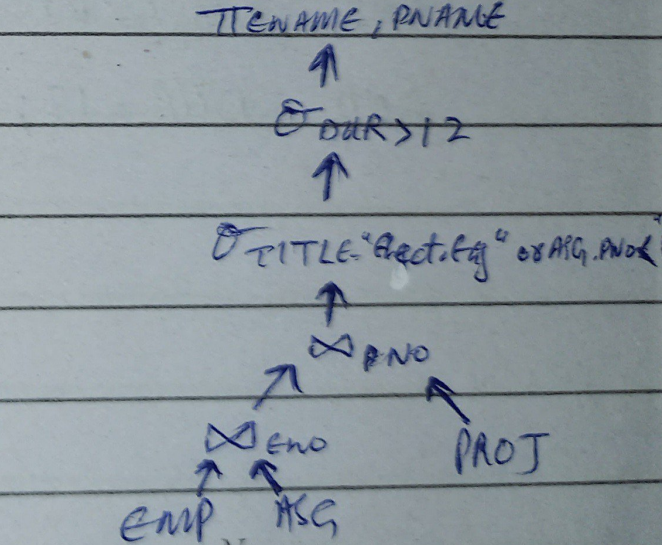
7.3 ⇒ Simplify query & transform into an optimized operator tree using restructuring algo (where selection & projection operations are applied).

```
SELECT ENAME, PNAME
FROM EMP, ASG, PROJ
WHERE (DUR > 12 OR RESP = "Analyst")
  AND EMP.ENO = ASG.ENO
  AND (TITLE = "Elect. Eng"
  OR ASG.PNO < "P3")
  AND (DUR > 12 OR RESP NOT = "Analyst")
  AND ASG.PNO = PROJ.PNO;
```

Signature _____

RC

⇒ After Simplify :-



No. _____

⇒ Generic Operator Tree.

Date

7.5 Simplifying :-

⇒ Consider only compound selection predicate on algebric exp :-

↳ ∨, ∧, ¬. ((NOT) EMP.ENO = ASG.ENO, ASG.PNO = PROJ.PNO.)

⇒ $\Pi_{ENAME, PNAME}(\sigma_{(DUR>12 \vee RESP="Analyst") \wedge EMP.ENO=ASG.ENO \wedge (TITLE="Elect.Eng" \vee ASG.PNO<"P3") \wedge (DUR>12 \vee \neg RESP="Analyst") \wedge ASG.PNO=PROJ.PNO}$.

$p_1 = DUR > 12$, $p_2 = RESP = "Analyst"$, $p_3 = TITLE = "Elect. Eng"$,
$p_4 = ASG.PNO < "P3"$.

⇒ The query qualification is :-

⇒ $(p_1 \vee p_2) \wedge (p_3 \vee p_4) \wedge (p_1 \vee \neg p_2)$.

• Acc to rules :-

Rule 1 : $p_1 \wedge p_2 \Leftrightarrow p_2 \wedge p_1$

⇒ $((p_1 \vee p_2) \wedge (p_1 \vee \neg p_2)) \wedge (p_3 \vee p_4)$

Rule 5 :- $p_1 \wedge (p_2 \vee p_3) \Leftrightarrow (p_1 \wedge p_2) \vee (p_1 \wedge p_3)$.

⇒ $(p_1 \wedge (p_2 \vee \neg p_2)) \wedge (p_3 \vee p_4)$

Rule 8 :- $p \vee \neg p \Leftrightarrow true$ Rule 3 : $p \wedge true \Leftrightarrow p$

⇒ $(p_1 \wedge true) \wedge (p_3 \vee p_4)$

⇒ $\boxed{p_1 \wedge (p_3 \vee p_4)}$

OR $(DUR > 12) \wedge (TITLE = "Elect. Eng." \vee ASG.PNO < "P3")$.

⇒ can't simplify further.

P.P(2023) Q3.a Date

4.2 Some architecture models favor the definition of a GCS (global conceptual schema), whereas other do not. What do you think? Justify your your selection with detailed technical arguments.

=> Global Conceptual Schema (GCS): Unified view of all data in a distributed system.

- Models favoring GCS: Centralized distributed dbs (eg: banking).
  - ↳ Ensures consistency, integrity & simpler queries.
- Models not favoring GCS: Multidatabase systems (eg: hospitals (autonomous sites))
  - ↳ sites are heterogenous & independent; global schema is hard to maintain.

↳ In Short: Use GCS when you need control & consistency; avoid it when flexibility & independence are more important.

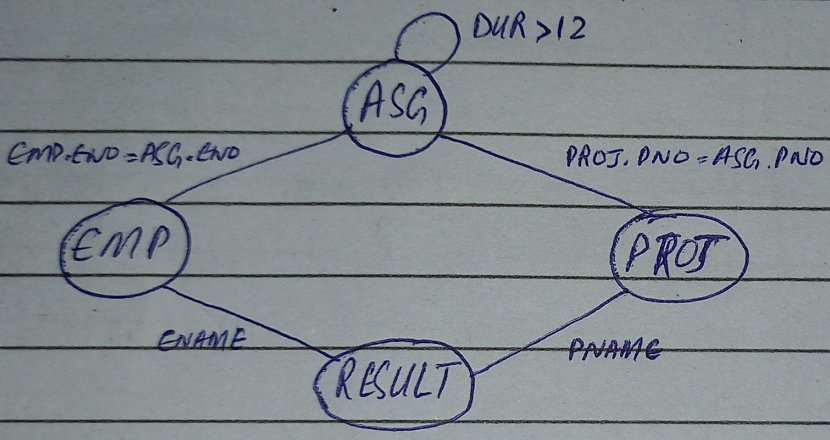
P.P(2023) Q3.b

Date

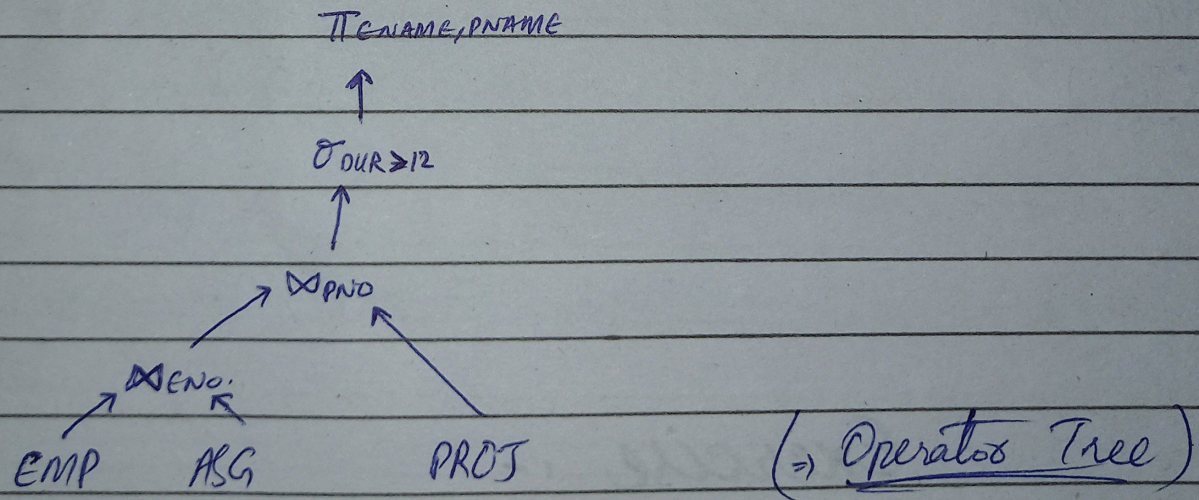
7.2 :- SQL Query :-

```
SELECT ENAME, PNAME
FROM EMP, ASG, PROJ
WHERE DUR > 12
AND EMP.ENO = ASG.ENO
AND PROJ.PNO = ASG.PNO
```

Give Query graph & map it into an operator tree.



(=> Query Graph)



(=> Operator Tree)

→ F.F (2023)

Q4.a

Date

Differentiate :-

1) LAV vs GAV:

- In LAV (Local-as-view) : GCS already exists, each local DB are views of a global schema. (based on global).
- In GAV (Global-as-view) : Build GCS by combing all local DBs. (build from local DBs).

2) Schema vs instance matching:

- Schema Matching : matching table names or column (attribute) names. (like "Name with "FullName")
- Instance Matching : matching actual data inside the tables. (like "Aliza" in one table & "Aliza Khan" in another).

3) Join vs Semi-Join :

- Join : A join combines two tables & shows data from both.

↳ Ex :

Table A

ID	Name
1	Alize
2	Sara

Table B

ID	City
1	Karachi
3	Lahore

query : $A \bowtie_{A.ID = B.ID} (B)$

ID	Name	City
1	Aliza	Karachi

- Semi-Joins:

↳ Only shows rows from the 1st table that have a matching in the second table.

↳ Ex - query : $A \ltimes_{A.ID = B.ID} (B)$ → only keeps rows from A that match B.

Table A

ID	Name
1	Aliza

→ P.P (2023)

4.5, 4.12

Q4.b

Given:

↳ GCS relations:

- Works (Id, Project) ⟶ employee works on a project.
- Area (Project, Field) → project belongs to a field.

↳ Databases:

DB1: Area (Id, field).

DB2: Teach (Professor, Course), In (Course, field).

DB3: Grant (Researcher, Grant NO), For (Grant NO, field).

(a) LAV mapping b/w DB1 & the GCS.

LAV = view over GCS.

⇒ DB1.Area (Id, field) ⟶ Works (Id, Project), Area (Project, field).

Date

(b) GLAV mapping b/w the GCS & local schemas.

GLAV = mix of LAV & GAV

→ 1st, GCS: (Work(Id, Project) & Area(Project, field)) as 1 GCS.

⇒ Works(Id, Project), Area(Project, field) → GCS(Id, field).

→ Now local schemas in GCS(Id, field) mapping.

⇒ GCS(Id, field) → DB1. Area(Id, field), DB2. Tech(Professor=Id Course), DB2. In(Course, field),
DB3. Grant(Researcher=Id, GrantNo), DB3. For(GrantNo, field).

(c) GAV mapping. one extra funds(GrantNo, Project), added in DB3.

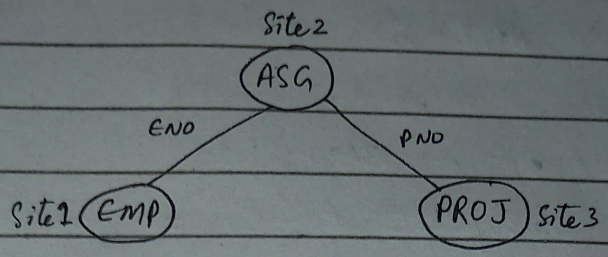
GAV = view over local

⇒ GCS. Works(Id, Project) → DB3. Grant(Researcher, GrantNo), DB3. funds(GrantNo, Project)
⇒ GCS. Area(Project, field) → DB3. For(GrantNo, field), DB3. funds(GrantNo, Project).

8.2:- → P.P (2023)

Q5.a

Consider join graph.



& following info:-

- size (EMP) = 100
- size (ASG) = 200
- size (PROJ) = 300
- size (EMP ⋈ ASG) = 300
- size (ASG ⋈ PROJ) = 200.

Based on func. of total transmission time describe an optimal join program.

⇒ Optimal Join Program (Minimizing Total transmission Time):

1) Step 1: Join ASG ⋈ PROJ at site 3. (ASG → PROR).
   - size (ASG ⋈ PROJ) = 200.

2) Step 2:- Join intermediate result with EMP at Site 3.
   - EMP ⋈ (ASG ⋈ PROJ)
   - size (EMP) ⋈ size (ASG ⋈ PROJ) = 100 + 200

⇒ Total transmission Time = 300

⇒ Always join tables producing smaller intermediates first & move the smaller to reduce transmission cost.

→ P.P (2023) Q5.b

8.4:- Consider 8.2 graph. & give a program that reduces each relation fully by semijoins.

-) Semijoin: $R \ltimes \{A\} S$ = means keep from R, only tuples whose A-values appear in S (matching).

- Site 1 → EMP(ENO)
- Site 2 → ASG (ENO, PNO)
- Site 3 → PROJ(PNO).

-) Reduce each relation using semijoin before the final join:-

- Step 1:- Reduce EMP (By removing those tuples which doesn't exist in ASG)
  ⇒ $EMP' = EMP \ltimes_{ENO} (ASG)$

- Step 2: Reduce PROJ (By removing those tuples which doesn't exist in ASG)
  ⇒ $PROJ' = PROJ \ltimes_{PNO} (ASG)$

• Step 3 :- Reduce ASG (By using Reduce EMP & PROJ by removing those ENO's & PNO's which doesn't match with ASG(ENO, PNO)).

⇒ $ASG' = (ASG \ltimes_{ENO} (EMP')) \ltimes_{PNO} PROJ'$

Finally :-

1) $EMP' \longrightarrow$ Site 2
2) $PROJ' \longrightarrow$ Site 2
3) Site 2 $\longrightarrow EMP' \ltimes_{ENO} ASG' \ltimes_{PNO} PROJ'$

⇒ Semijoin = reduce the size of each relation before sending or joining. Saves communication & computation.