2 and 3 marks and 5 marks Qtr. Upto this.

The entire relation also has a security level which is the lowest level of any data it contains.
for example, if PROJ* has security level 'C'.
A relation can then be accessed by any subject having a security level which is the same or higher.
However, a subject can only access data for which it has clearance.
Thus, attributes for which a subject has no clearan will appear to the subject as null values with associated security level which is the same as the subject.

Confidential relation: PROJ*

PNO	SL1	PNAME	SL2	Budget	SL3	LOC	SL4
P1	C	Instrumentation	C	150000	C	Montreal	C
P2	C	DB Devel	C	null	C	null	S

* Distributed access control:-
The additional problems of access control in distributed environment originate from the fact that objects and subjects are distributed and the messages with sensitive data can be read by unauthorized users.
These problems are
(1) Remote user authentication.
(2) Management of discritionary access rules.
(3) Handling of views and of user groups.

(4) enforcing multilevel access control.

* Three solutions for possible for managing authentication.

(1) Authentication information maintained at a central site for global users. (problem can be single point of failure)

(2) The information for authenticating users (user name & password) is replicated at all sites in the catalog.
(Solution 2 is more costly in terms of directory management).

(3) All sites of the DDBMS identify & authenticate themselves similar to the way users do.

Covert channels
. Indirect means to access unauthorized data.

Semantic integrity control:
. Maintain database consistency by enforcing a set of constraints defined on the database.

Two main types of integrity constraints:

1. Structural constraints → express basic semantic properties of inherent model. Examples of such constraints are:
- unique key constraints in relational model.
or one-to-many associations.

Learn triggers

2. Behavioural constraints : regulate application behaviour.
E.g:- dependencies in Relational model.

19/19

Enforcing Integrity constraints: is costly because it generally requires access to a large amount of data that are not directly introduced in the database updates

Various solutions for enforcing integrity constraints:

(1) Limit the number of constraints that need to be enforced.
(2) decrease the number of data access to enfose a given constraint in the presence of an update transaction.
(3) Define prevention strategy that detect inconsistencies in a way that avoids undoing updates.
(4) perform as much integrity control as possible at compile time.

Centralized semantic Integrity ~~Constraints~~ control:

(1) Specification of integrity constraints:
Integrity constraints should be manipulated by the database administrator using a high-level language - SQL

Examples of integrity constraints will be given on the following database.

EMP (ENO, ENAME, TITLE)
PROJ (PNO, PNAME, BUDGET)
ASG (ENO, PNO, RESP, DUR)

1. Employee number in relation EMP cannot be null

ENO NOT NULL IN EMP.

2. The pair (ENO, PNO) in the unique key in the relation ASG.

(ENO, PNO) UNIQUE IN ASG

3. The project number PNO in relation ASG is foreign key matching the primary key PNO of relation PROJ.

PNO IN ASG REFERENCES PNO IN PROJECT

4. The employee number functionally determine the employee name.

ENO IN EMP DETERMINES ENAME.

② Integrity Enforcement:

Enforcing semantic integrity that consists of rejecting update transactions that violate some integrity constraints

Two methods

1. prevention of inconsistencies:
An update is executed only if it changes the database state to a consistent state.

2. Detection of inconsistencies:
The update transaction 'u' is executed, causing a change of database state from D to D'.
- If state D' is inconsistent, then DBMS may reach another consistent state D'', by modifying D' or to restore state D by undoing u.

16/9/19

* Distributed Semantic Integrity Control

Individual constraints:- Single-relation single variable constraints.

A constraint C is not compatible with a fragment predicate P if "C is true" implies that "p is false" and is compatible with P otherwise.

Example:- Consider, Relation EMP, horizontally fragmented across 3 sites using predicates.

~~P1: 0 ≤ EMP < E3~~ P1: 0 ≤ ENO < E3
~~P2: E3 ≤ EMP ≤ E6~~ P2: E3 ≤ ENO ≤ E6
~~P3: EMP > E6~~ P3: ENO > E6

and the domain constraint

C : END < E4

Constraint C is compatible with P1. (if C is true, P1 is true) and P2 (if C is true P2 is not necessarily false) but not with P3 (if C is true, then P3 is false).

⇒ ∴ C should be rejected, because the tuples at site 3 cannot satisfy C.
Hence EMP does not satisfy C.

* Query Processing :-

Query processing in DDBMS.

↓ high level user query (SQL)

↓

Query Processing

↓

Low level data manipulation commands for DDBMS. (Relational algebra)

The main function of query processing is to transform a high level query (SQL) into an equivalent lower-level query (relational algebra).

Example :- consider subset of the engineering database schema.

EMP(ENO, ENAME, TITLE)
ASG(ENO, PNO, RESP, DUR)

"Find the names of employees who are managing a project".

In SQL,

```
Select ENAME
from EMP, ASG
where EMP.ENO = ASG.ENO
      AND RESP = "Manager".
```

The two equivalent relational algebra expressions are

$$\Pi_{ENAME}\left(\sigma_{RESP=Manager \wedge EMP.ENO=ASG.ENO}(EMP \times ASG)\right).$$

and

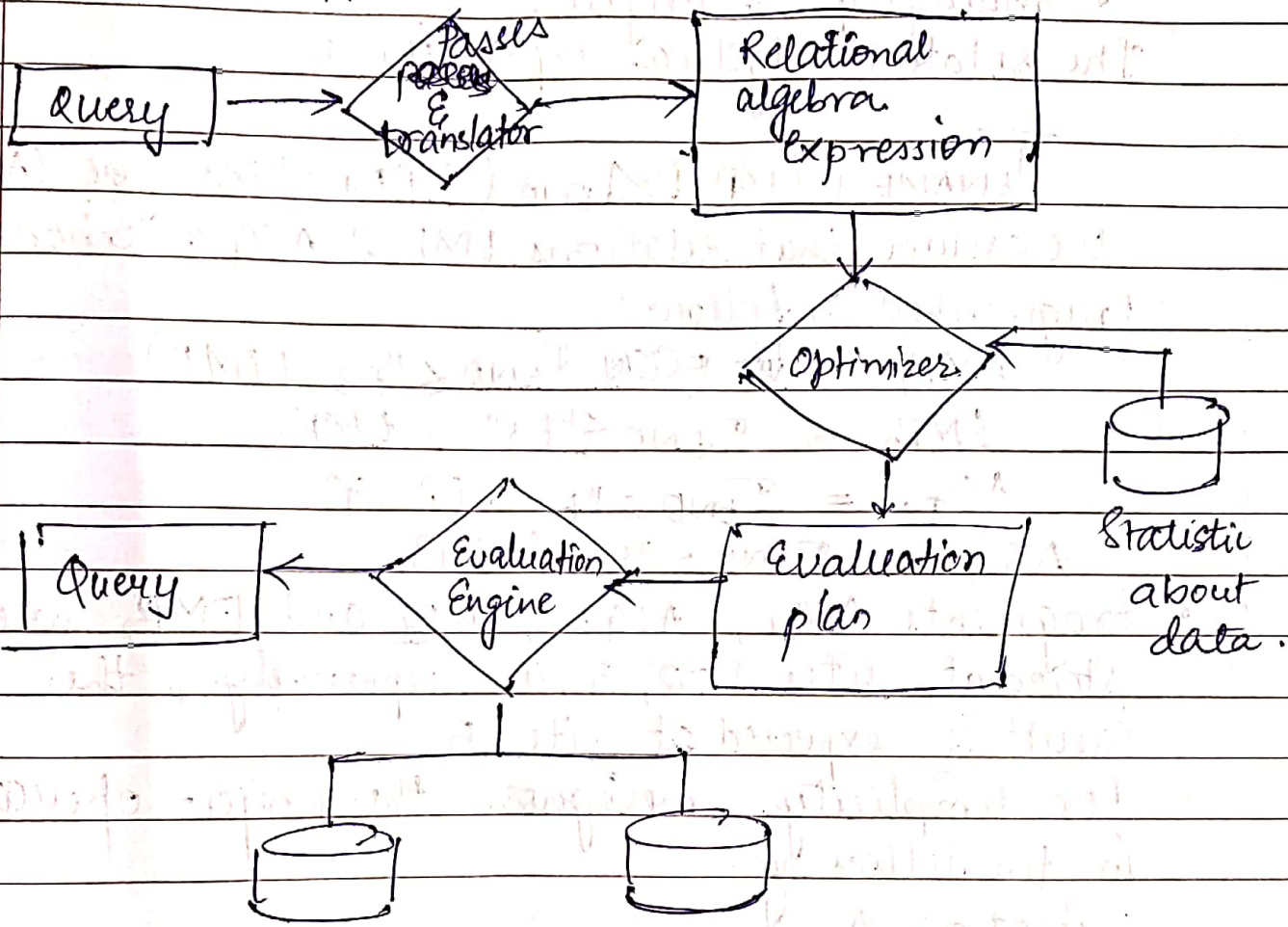
$$\Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{RESP=Manager}(ASG))).$$

Second query, which avoids the cartesian product of EMP & ASG, consumes less computing resources than first, & hence should be retained.

28/9/19.

Basic steps in Query Processing:-

(1.) Passing & Translation.
(2.) Optimization
(3.) Evaluation.



Passes & translator:- Translate the query into its internal form. This is then translated into relational algebra.
- passes checks syntax, verifies Relation.

- Evaluation: The query execution engine takes a query evaluation plan, executes that plan & return the answer to the query.
- Query Optimization:- Amongst all equivalent evaluation plans choose the one with lowest cost.

Eg:- Find the names of employees who are

managing a project. The relation algebra expression is

Eg:- Find the names of employees who are managing a project.

The relation algebra expression is

$$\Pi_{ENAME}(EMP \bowtie_{ENO} (\sigma_{RESP = "Manager"}(ASG)))$$

we assume that relations EMP & ASG are horizontally fragmented as follows:

$EMP_1 = \sigma_{ENO \leq "E3"}(EMP)$

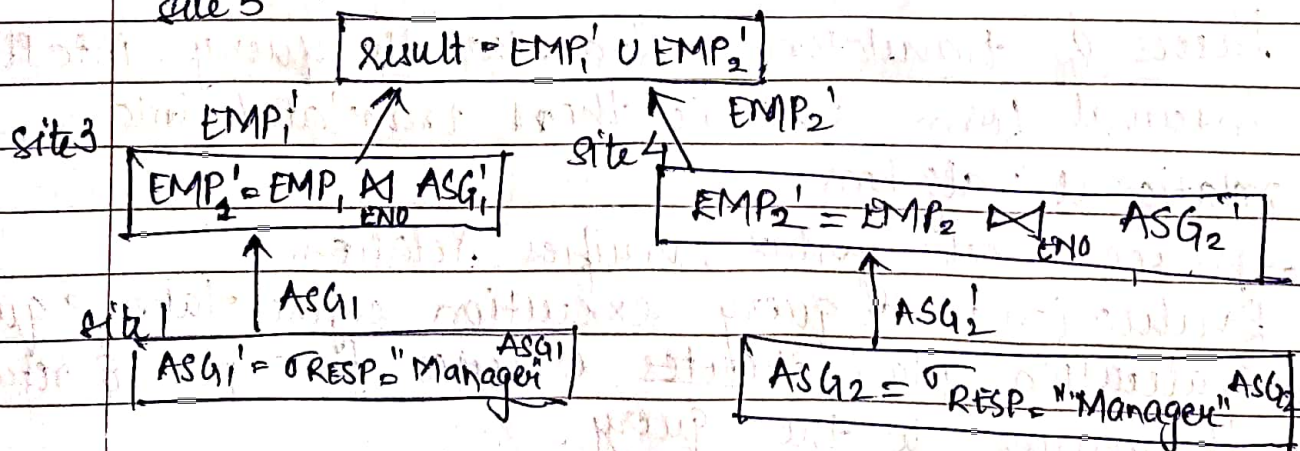
$EMP_2 = \sigma_{ENO < "E3"}(EMP)$

$ASG_1 = \sigma_{ENO \leq "E3"}(ASG)$.

$ASG_2 = \sigma_{ENO > "E3"}(ASG)$.

- Fragments ASG_1, ASG_2, EMP_1 and EMP_2 are stored at sites 1, 2, 3, 4. respectively & the result is expected at site 5.
- For simplicity, we ignore the project operator in the following :-

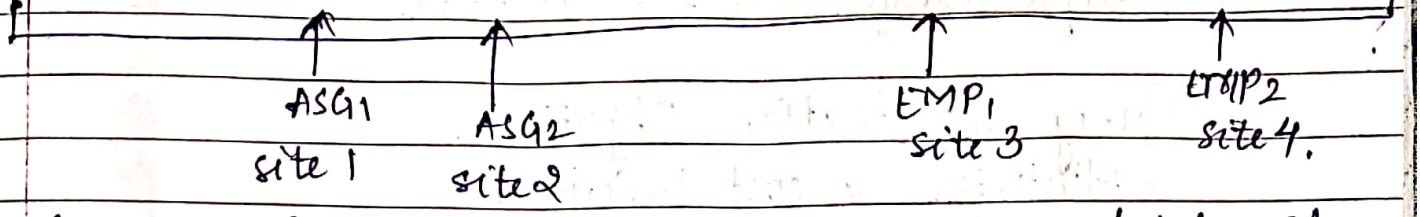
Strategy A.



Strategy B.

site B

result = (EMP, ∪ EMP2) ⋈ ENO (RESP = "Manager" (ASG, ∪ ASG2))



- Strategy B centralizes all the operand data at the result site before processing query.

→ Assume that a tuple access, denoted by tupacc is 1 unit, the tuple transfer is denoted by tuptrans in 10 units.

- Assume that the relation EMP and ASG have 100 & 1000 tuples:

(1) Produce ASG' by selecting ASG requires (10+10) * tupacc = 20.

(2) Transfer ASG' to the site of EMP (10+10) * tuptrans = 200.

(3) Produce EMP' by joining ASG' and EMP requires (10+10) * tupacc * 2 = 40.

(4) Transfer EMP' to result site requires (10+10) * tuptrans = 200.

∴ total cost of strategy A = 460.

(1) Transfer EMP to site 5 requires 400 * tuptrans = 4000.

(2) Transfer ASG to site 5 requires 1000 * tuptrans = 10,000.

(3) Produce ASG' by selecting ASG requires 1000 * tupacc = 1000.

(4) Join EMP and ASG' requires 400 * 20 * tupacc = 8000.

∴ total cost of strategy B = 23000.

Assume.

Size (EMP) = 400, size (ASG) = 1000, 20 tuples with TITLE = "Manager" tuple access cost = 1 unit, tuple transfer cost = 10 units.

Query Optimization Objectives.

- Minimize a cost function.
  I/O cost + CPU cost + communication cost.
- Maximize throughput.

Complexity of Relational Algebra operations.

- Measured by cardinality n & tuples are sorted on comparision attributes.

Operation	Complexity.
Select, project (without duplicate elimination)	O(n)
project (with duplicate elimination groups).	O(nlogn).
Join, semijoin, Division, set operator.	O(nlogn)
Cartesian product.	$O(n^2)$.

Query optimization Issues.

(1) Types of Optimization.

(a) Exhaustive search: Exhaustively predict the cost of each strategy &

Select the strategy with minimum cost.
- cost based.
- optimal
- workable for small solution spaces.

(ii) Heuristics :- It is used to avoid high cost of exhaustive search. The purpose of heuristic in to reduce the solution space.

(2) optimization granularity.
- single query at a time.
- Multiple query at a time.

(3) optimization timing.

Query optimization Issues.
- Static : Do it at compile time
- Dynamic : Do it at execution time.
- Hybrid : compile using a static algorithm
  If the errors in estimate size, reoptimizing at run time.

26/9/19

(4) a. Statistics :- To minimize the probability of error, more detailed statistic such as histograms of attribute values are sometimes used at the expenses of higher management cost.

(5) Decision sites :-
Query optimization may be done by
- Single site :- centralized approaches.

- single site determines the best schedule
  - simple.
  - Need knowledge about the entire distributed database.
- All the sites involved - distributed approach.
  - coopeuation among sites to determine the schedule.
  - Need only local information.
  - Hybrid - one site makes major decision in cooperation with other sites making local decisions.

(6) Exploitation of the N/w Topology :-

(i) Wide Area Network (WAN)

(ii) - communication cost will be dominate; ignore all other cost factors.

(ii) Local Area Network (LAN).

- communication cost not that dominates.

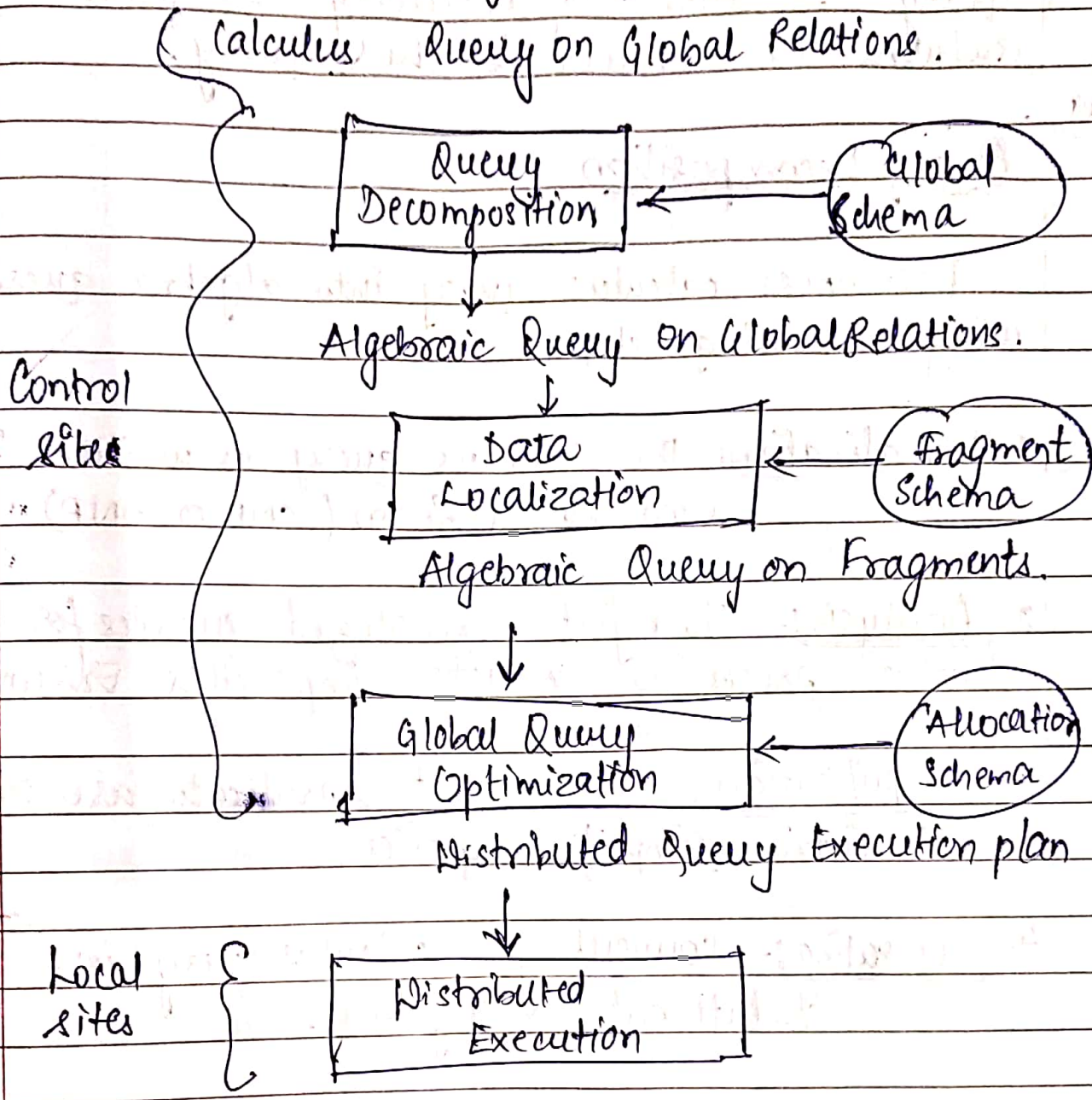
(7) Exploitation of Replicated Fragments.

- using replications to minimize communication costs.

(8) Use of Semijoins.

- using semijoins to reduce the size of operand relations.

Layers of Query Processing.



Four main layers are involved in distributed Query processing.

The first 3 layers (query decomposition, data localization, & global query optimizatn). map input query into an optimized distributed query execution plan.

Query decomposition & data localizatn correspond to query rewriting.

The ~~fourth~~ fourth layer performs distributed query execution by executing the plan & returns the answer to the query.

30/09/19

Query Decomposition

Decompose calculus query into algebra query using following steps:

1. Normalization: The calculus query is written in a normalized form (CNF or DNF)

2. Analysis:- To reject normalized queries for which further processing is either imposible or unnecessary

3. Simplification:- Redundant predicates are eliminated to obtain simplified queries.

4. Rewriting:- Converting a calculus query in relational algebra.

(i) Normalization:-

Example:- Find the names of employees who have been working on Project P_1 for 12 or 24 months.

```
Select ENAME
From EMP, ASG
Where EMP.ENO = ASG.ENO AND ASG.PNO="P1"
      AND (DUR = 12 OR DUR = 24)
```

The CNF is

EMP.ENO = ASG.ENO ∧ ASG.PNO = "P1" ∧
(DUR = 12 OR DUR = 24).

The DNF is

(EMP.ENO = ASG.ENO ∧ ASG.PNO = "P1" ∧
DUR = 12) ∨ (EMP.ENO = ASG.ENO ∧ .....).

2. Analysis :-

Example :- Consider the following SQL query.

Select E# ← undefined attribute
From EMP
where ENAME > 200 ← type mismatch
This query is incorrect.

Query graph. : 2 kinds of nodes.
- one node represents the result relation.
- other node represents operand relations.

2 types of edges:

- An edge to represent a join if neither of its 2 nodes in the result.
- An edge to represent a projection if one of its node in the result node.

Nodes & edges may be labelled by prediction

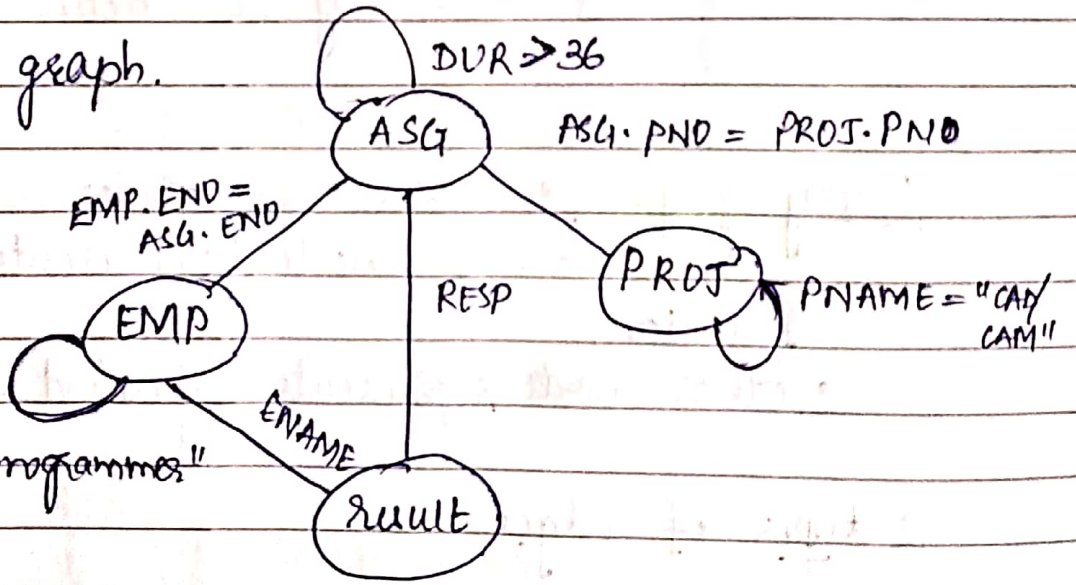
for selection, projection or join.

Examples:

Find the names & responsibilities of programmers who have been working on the "CAD/CAM" project for more than 3 years.

```
Select ENAME, RESP
from EMP, ASG, PROJ
Where EMP.ENO = ASG.ENO AND ASG.PNO = PROJ.
                                      PNO,
      AND PNAME = "CAD/CAM" AND DUR > 36
      AND TITLE = "Programmer".
```

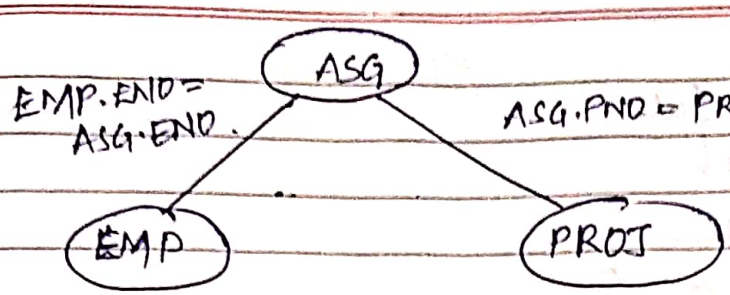
Query graph.



Join graph :- An important subgraph of the query graph, in which only joins are considered. It will be useful in query optimization.

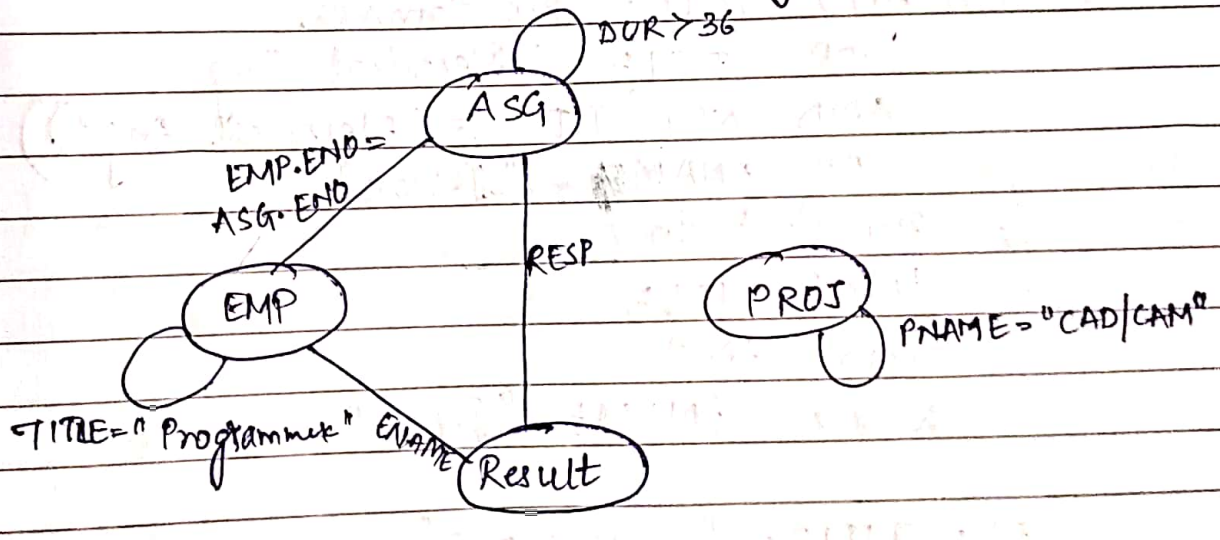
8/10/19.

format → NoFil
No Outline.



Example:- Consider the following SQL query

```
Select ENAME, RESP
from EMP, ASG, PROJ
Where EMP.END= ASG.END.
      AND PNAME = "CAD/CAM"
      AND DUR >= 36
      AND TITLE = "Programmer".
```



The query graph is disconnected, which many that query is semantically incorrect.

(3) Simplification:- Using idempotency rules to eliminate redundant predicates from the WHERE clause

(i) $P \wedge P \iff P$

(ii) $P \vee P \iff P$

(iii) $P \wedge true \iff P$

(iv.) $P \vee false \iff P$.

(v) $P \wedge false \iff false$

(vi) $P \wedge \neg P \iff false$

(vii) $P \vee \neg P \iff true$.

(viii) $P_1 \wedge (P_1 \wedge P_2) \iff P_1$

(ix) $P_1 \vee (P_1 \wedge P_2) \iff P_1$

Simplification Example.

```
Select TITLE
From EMP
where (NOT (TITLE = "programmer")
      AND (TITLE = "programmer"
        OR TITLE = "Electrical Eng")
      AND NOT (TITLE = "Electrical Eng"))
        OR ENAME = "J. Doe"
```

is equivalent to

```
Select TITLE
From EMP
where ENAME = "J. Doe".
```

$P1$ = TITLE = "Programmer"

$P2$ = TITLE = "Electrical Eng"

$P3$ = ENAME = "J. Doe"

The ~~quality~~ query qualification is

$(\neg P1 \wedge (P1 \vee P2) \wedge \neg P2) \vee P3$

$= (\neg P1 \wedge P1 \wedge \neg P2) \vee (\neg P1 \wedge P2 \wedge \neg P2) \vee P3$ (DNF)

$= (false \wedge \neg P2) \vee (\neg P1 \wedge false) \vee P3$

= false ∨ false ∨ P3

= P3.

4. Rewriting : Converting a calculus query in relational algebra expression.
   - making use of query tree.

query tree :- It is defined by
- a root node representing the query result.
- leaves representing relations.
- non-leaf nodes representing relations produced by operations.
- edges from leaves to root representing the sequence of operations.

Example :- Find the names of employees other than Jhon who worked on the CAD/CAM project for either one or two years.

The SQL expression.

Select ENAME

From

2) Reduction for vertical fragmentation.

The VF function distributes a relation based projection attributes.

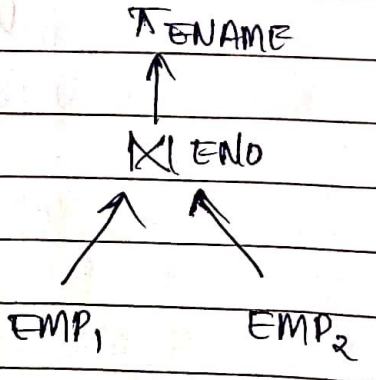
Ex 4: The relation EMP can is divided into 2 vertical fragmentation. fragments.

$$EMP_1 = \Pi_{ENO,ENAME}(EMP)$$

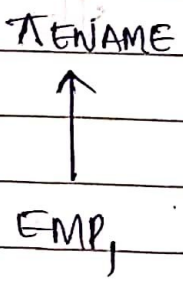
$$EMP_2 = \Pi_{ENO,TITLE}(EMP)$$

Consider the following SQL Query.

```
select ENAME
from EMP
```



Initial Tree



Reduced Tree

3) Reduction for Derived horizontal fragmentation. (DHF)

DHF is another way of distributing 2 relations so that the joint processing of select & join is improved.

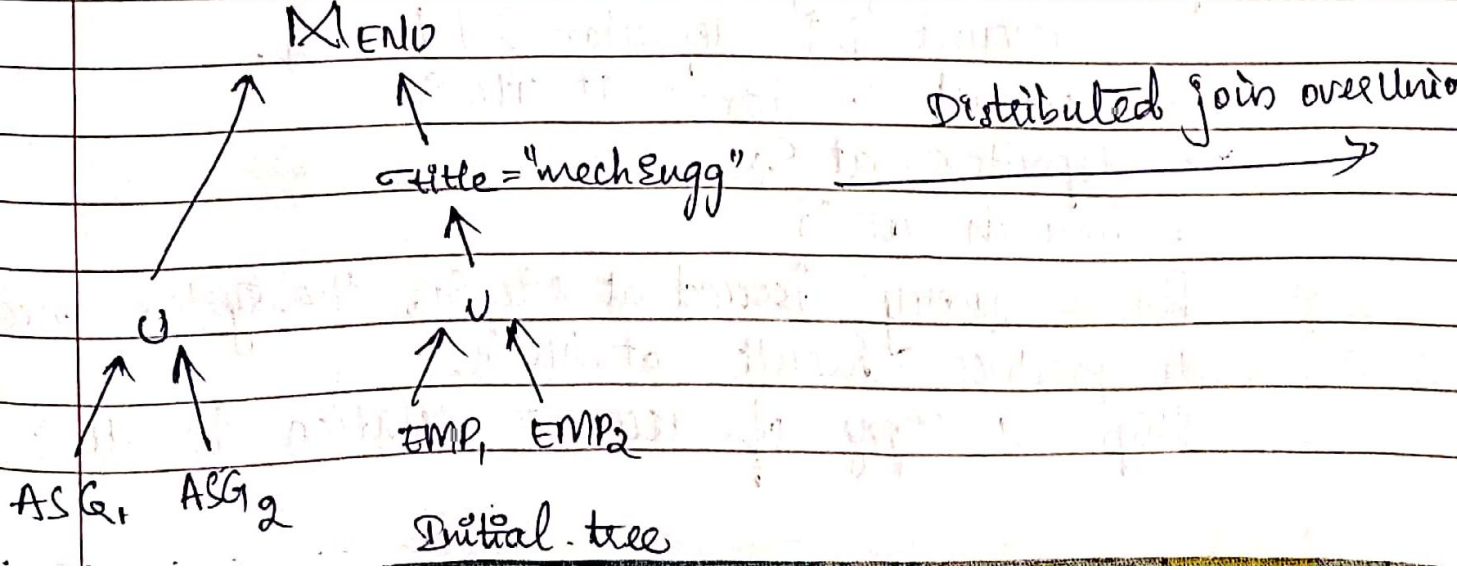
Example:- Relation ASG (ENO, PNO, RESP, DUR) can be indirectly fragmented as follows.

$ASG_1 = ASG \ltimes_{ENO} EMP_1$,
$ASG_2 = ASG \ltimes_{ENO} EMP_2$
$ASG = ASG_1 \cup ASG_2$

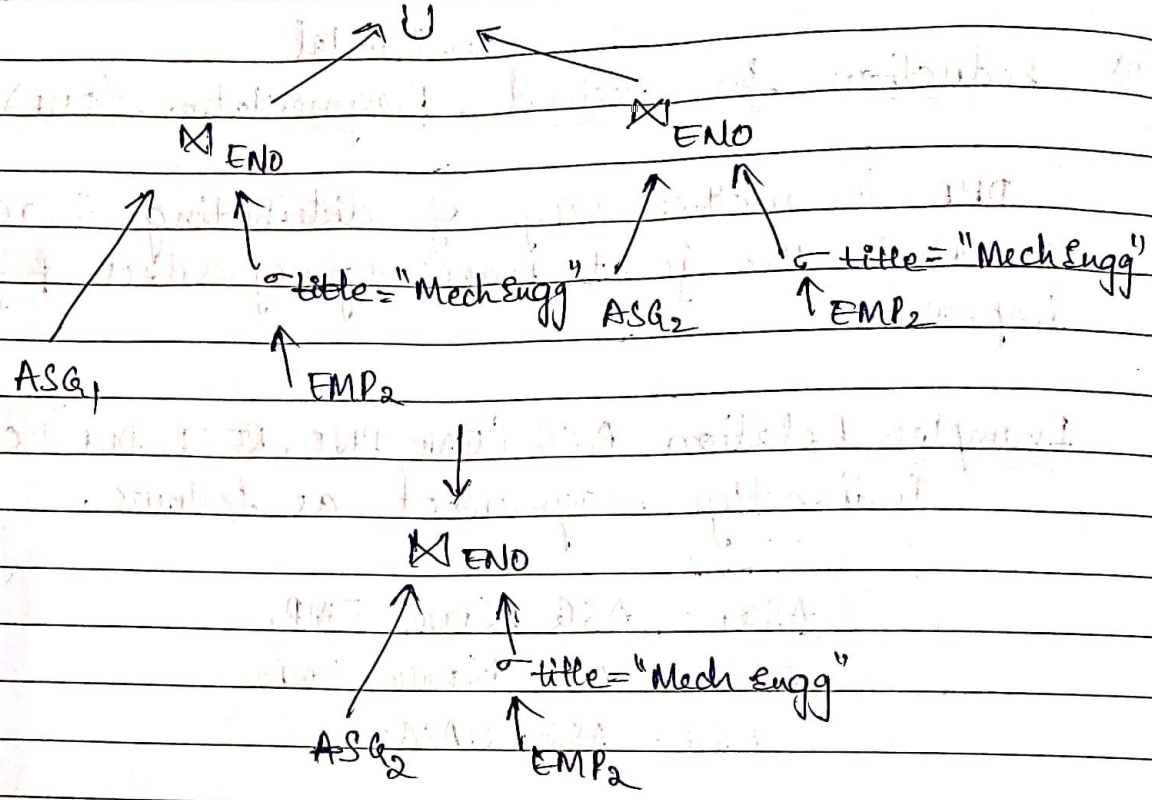
$EMP_1 = \sigma_{TITLE = \text{"programmer"}}(EMP)$
$EMP_2 = \sigma_{TITLE = \text{"programmer"}}(EMP)$.

Consider SQL query.

```
Select *
From EMP, ASG
where ASG.ENO = EMP.ENO
AND EMP.Title = 'MECH ENGG'.
```



Initial tree



Reduced Tree.

11/10/19

Optimization of Distributed Queries

Simple join processing.

Consider the following relational algebra expression in which the 3 relations are neither replicated nor fragmented.

account ⋈ depositor ⋈ branch.

- account is stored at site S_1
- depositor at S_2
- branch at S_3

For a query issued at site S_1, the system needs to to produce result at site S_1.

Ship a copy of account relation to site S_2 &

compute $temp_1$ = account ⋈ depositor at S_2.
Ship $temp_1$ from S_2 to S_3 & compute
$temp_2$ = $temp_1$ ⋈ branch at S_3

Ship the $temp_2$ to S_1

Must consider following factors.
1. Amount of data being shipped.
2. cost of transmitting data block b/w sites.
3. Relation processing speed at each site.

Semi-join Strategy:-

Suppose that we wish to compute
$r_1 \bowtie r_2$
where r_1 be a relation with schema R_1 stored at site S_1 & r_2 be a relation with schema R_2 stored at site S_2.
Evaluate r_1 ⋈ r_2 and obtain result at S_1.

1. Compute $temp_1 \leftarrow \Pi_{R_1 \cap R_2}(r_1)$ at S_1
2. Ship $temp_1$ from S_1 to S_2.
3. Compute $temp_2 \leftarrow r_2 \bowtie temp_1$ at S_2
4. Ship $temp_2$ from S_2 to S_1.
5. Compute $r_1 \bowtie temp_2$ at S_1. This is same as $r_1 \bowtie r_2$.

Join Strategies that exploit parallelism.

Consider $r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4$, where relation r_i is stored at site S_i. The result must be presented at site S_1.

1. R_1 is shipped to S_2 & $r_1 \bowtie r_2$ is computed at S_2 simultaneously R_3 is shipped to S_4 & $r_3 \bowtie r_4$ is computed at S_4.

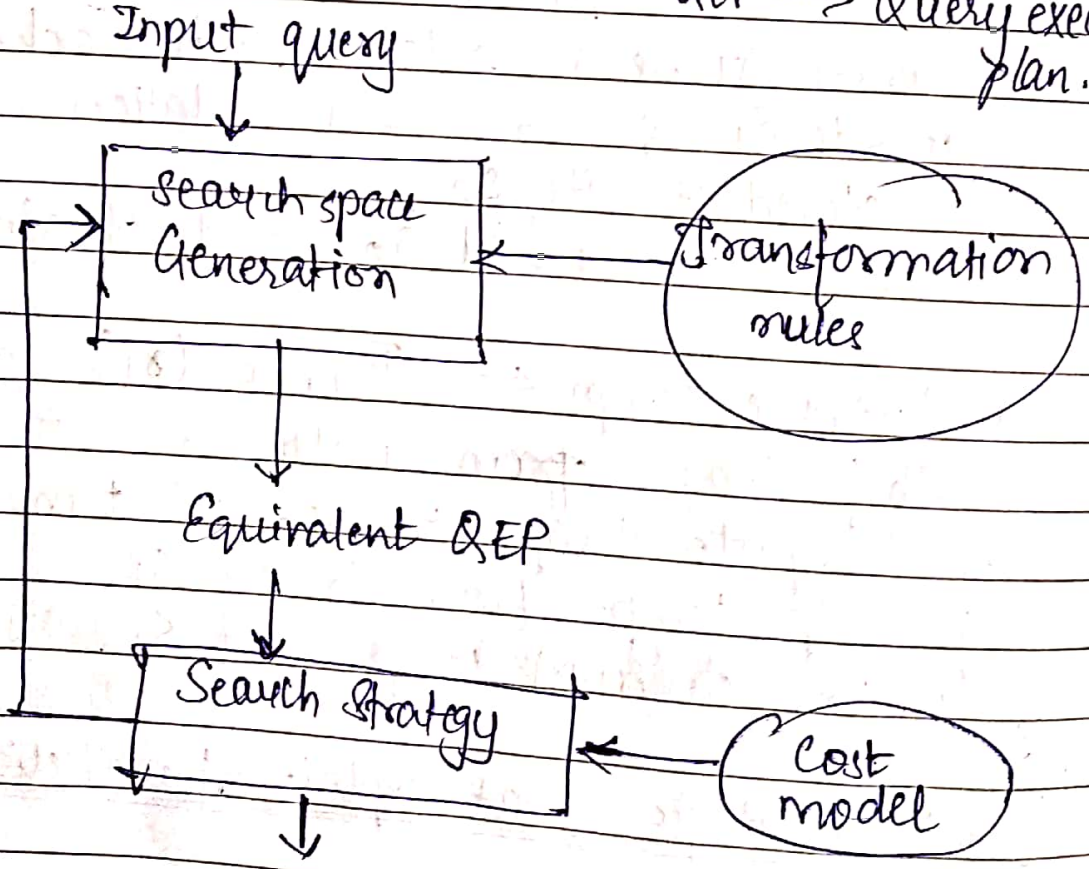
2. S_2 ships tupples of $(r_1 \bowtie r_2)$ to S_1 and S_4 ships tupples of $(r_3 \bowtie r_4)$ to S_1.

3. Once tuple of $(r_1 \bowtie r_2)$ & $(r_3 \bowtie r_4)$ arrive at S_1. $(r_1 \bowtie r_2) \bowtie (r_3 \bowtie r_4)$ is computed ~~parallelly~~ in parallel with the computation of $(r_1 \bowtie r_2)$ at S_2 and the computation of $(r_3 \bowtie r_4)$ at S_4.

14/10/19

Query optimization process

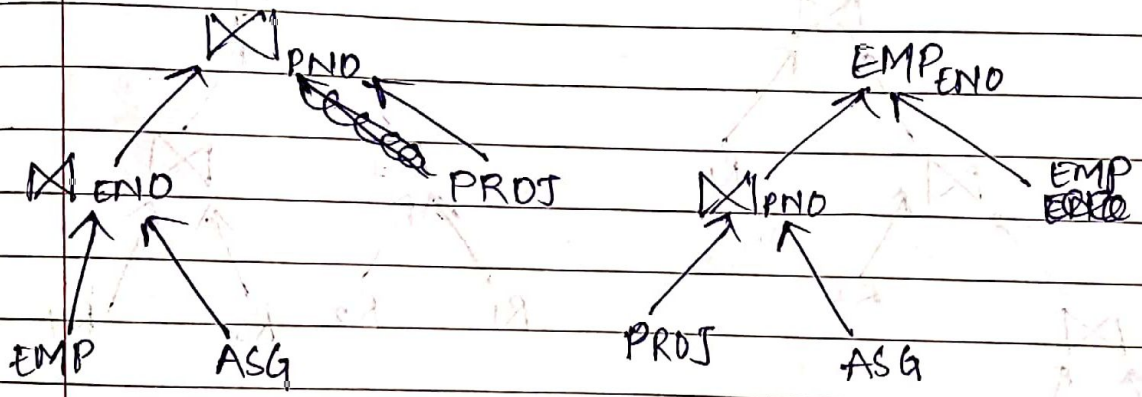
QEP → Query execution plan.

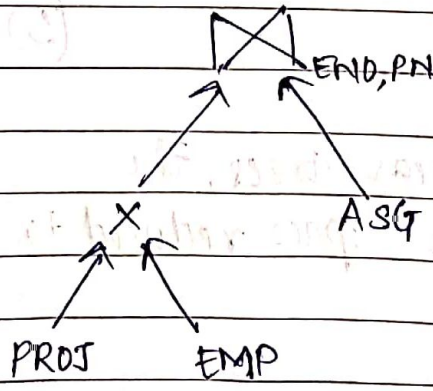


Query execution plans are typically done by means of operator trees, which define the order in which operations are executed.
For a given query, the search space can then be defined as the set of equivalent trees that can be produced using transformation rules.

Example:- Consider the following query.

```
Select ENAME, RESP
From EMP, ASG, PROJ
Where EMP.ENO = ASG.ENO
AND ASG.PNO = PROJ.PNO.
```





$(EMP \bowtie ASG) \bowtie PROJ = EMP \bowtie (ASG \bowtie PROJ)$

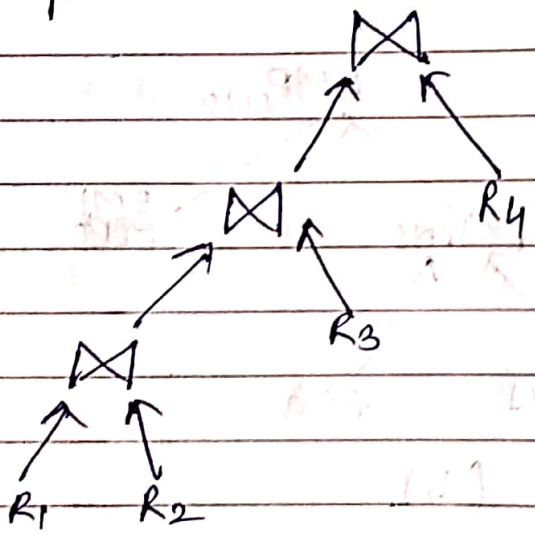
For N relations, there $O(N!)$ equivalent join trees that can be obtained by applying commutativity &

associativity rules.

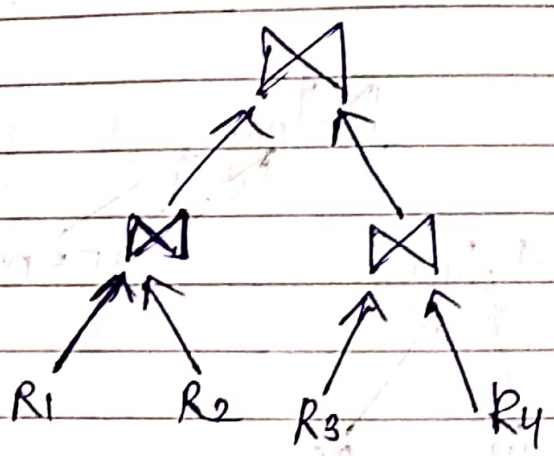
Search space: Restrictions.

Restrict - the shape of the join trees, using 2 kinds of join trees: linear & vs bushy trees.

A linear tree is a tree such that at least one operand of each operator node in a base relation. A bushy tree is more general and may have operators with no base relations as operands.



(a) linear join tree



(b) bushy join tree

By considering only linear trees, the size of the search space reduced to

$O(2^n)$

Search Strategy :-

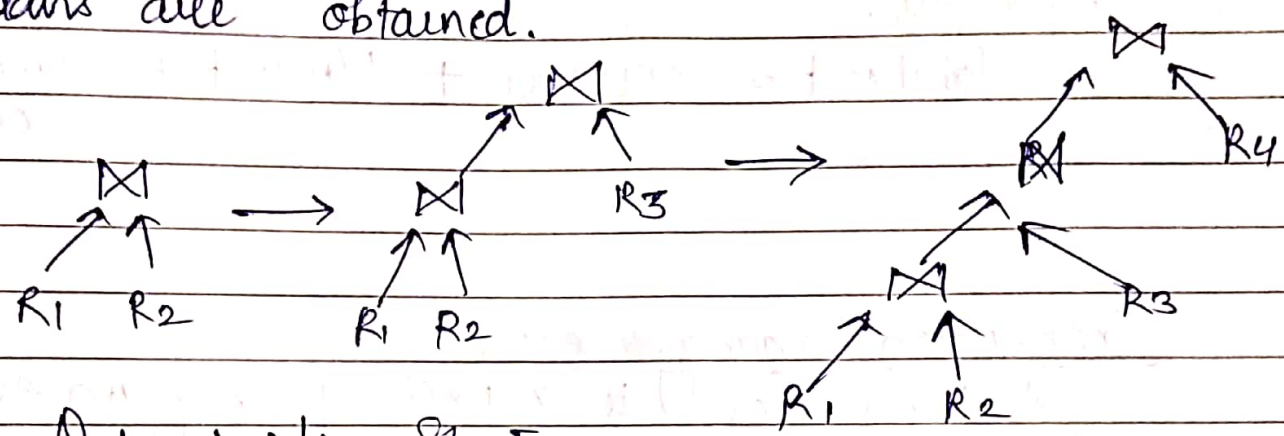
The most popular search strategy used by query optimizers are deterministic & randomized

40 Exam

Hrs. 10
10
40
60.

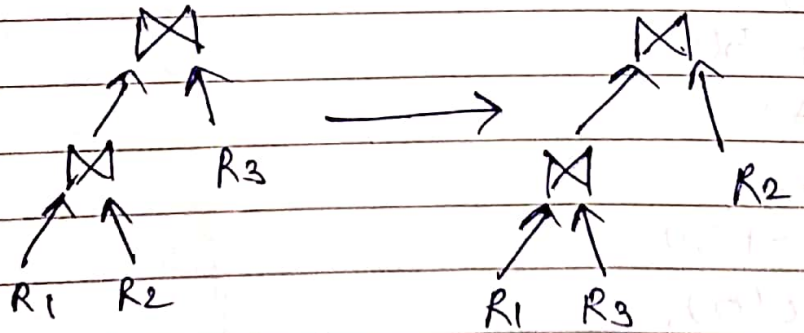
Strategies.

~~Determination~~ Deterministic strategies proceed by building plans, starting from base relations, joining one more relations at each step until complete plans are obtained.



Deterministic Strategy.

Randomized Strategy: search for ~~optimization~~ optimalities around a particular start point.



15/6/19

Distributed cost functions.

The cost of a distributed execution strategy can be expressed w.r.t either the total time or the response time.

The total time is the sum of all time components.

Total cost = CPU cost + I/O cost + Communicatn cost

Transaction management.

A transaction is a unit of program execution that accesses & possibly updates various data items.

Example:- Transaction to transfer $50 from account A to account B.

(1) Read (A)
(2) A ← A - 50
(3) Write (A)
(4) read (B)
(5) B = B + 50
(6) write (B).

Two main issues to deal with :-

- Failures of various kinds :- such as h/w failures s/w failures & system crash.
- Concurrent execution of multiple transactions.

Properties of transactions (ACID):-

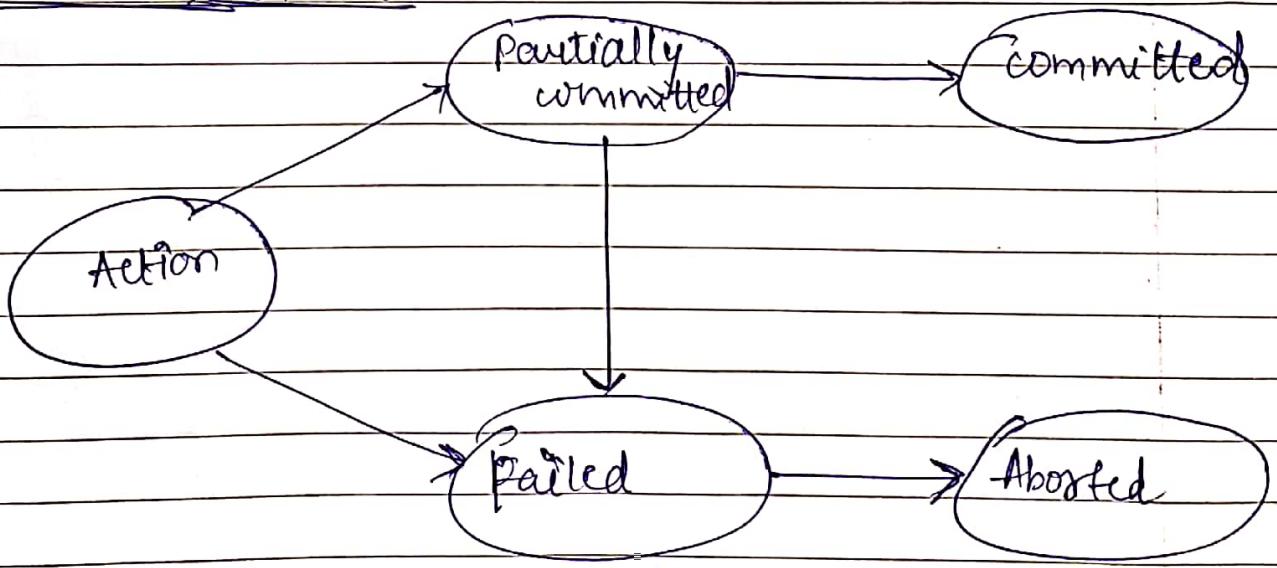
(1.) **Atomicity**:- Either all or none of transactions op^{n}s are completed.

(2.) **Consistency**:- No violation of integrity constraints.

(3.) **Isolation**:- Although multiple transactions may execute concurrently, each transaction's must be unaware of other concurrently executing transactions.

(4.) **Durability**:- After transaction completes successfully, the changes it has made to the dB ~~xxxxx~~ never lost, inspite of system failures.

Transaction State:-



Action:- the initial state; the transaction stays in this state while it is executing.

Partially committed:- after the final statement has been executed.

Failed state:- ~~After the discovery~~ that normal execution can no longer proceed.

Aborted :- after the transactn has been rolled back & dB restored to its state prior to the start of the transaction.

Committed :- after succesfull completion of transaction.