Q1. Differentiate Horizontal and vertical partitions?

**Vertical Fragmentation:**
Different subsets of attributes are stored at different places, like, Table EMP (eId, eName, eDept, eQual, eSal)
Interests of the local and head offices may result following vertical partitions of this table: EMP1 (eId, eName, eDept)
EMP2 (eId, eQual, eSal)

**Horizontal Fragmentation:**
It is based on the localization of data rows of a table are split on multiple sites, like the data in CLIENT(cAC#, cName, cAdr, cBal)
Table is placed in different databases based on their location, like from Lahore, Pindi, Karachi, Peshawar, Quetta

Q2. Differentiate round robin, hash partitioning and range partitioning using diagrams?

**Round Robin:**
With n partitions, the i^{th} tuple is placed in partition \((i \mod n)\). This strategy enables sequential access to a relation to be done in parallel. The direct access to individual tuples, based on predicate, requires accessing the entire relation as shown in figure 4.
Range Partitioning:

It distributes tuples based on value intervals of some attribute as shown in figure 6. It is suitable for exact match and range queries. Range partition can result in high variation in partition size.

Hash partitioning:

It applies a hash function to some attribute which yields the partition number as shown in figure 5. This strategy allows exact-match queries on the selection attribute to be processed by exactly one node and all other queries to be processed by all the nodes in parallel.
Q3. Comparison of Pessimistic and optimistic techniques.

The concurrency control mechanisms can be grouped into two broad classes: Pessimistic concurrency control methods and Optimistic concurrency control methods. Pessimistic algorithms synchronize the concurrent execution of transactions early in their execution life cycle, whereas optimistic algorithms delay the synchronization of transactions until their termination.

The pessimistic group consists of locking based algorithms, ordering (or transaction ordering) based algorithms, and hybrid algorithms. The optimistic group can, similarly, be classified as locking-based or timestamp ordering-based.

In the locking-based approach, the synchronization of transactions is achieved by employing physical or logical locks on some portion or granule of the database. The size of these portions, usually called locking granularity is an important issue.

The timestamp ordering (TO) class involves organizing the execution order of transactions so that they maintain transaction consistency. This ordering is maintained by assigning timestamps to both the transactions and the data items that are stored in the database. These algorithms can be Basic TO, Multiversion TO, or Conservative TO.

**Pessimistic & Optimistic.**

**Pessimistic** approach synchronizes transactions early • Locking-based
  o Centralized Locking
  o Primary Copy Locking
  o Distributed Locking-
  • Timestamp Ordering (TO)
    o Basic TO
    o Multiversion TO
    o Conservative TO
  • Hybrid
Optimistic do this late in execution life cycle of transactions

- Locking-based
- Timestamp ordering-based

Q4. Comparison of array processing and multiprocessing with diagram. 15 marks.

Parallel Processing basics:-
1: Multiprocessor Systems
Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system.
One computer with a global RAM shared by many processors and are tightly coupled. In multiprocessor, thousands of processor is controlled by single operating system, and each processor accessing the same job queue.

Therefore, Synchronization and interprocessor communication is required. Interconnection systems are
- Time shared or common bus
- Crossbar switches
- Multiport-memory systems
- Multistage networks

2: Associative Processors
They are designed to speed up the search of data items in memory. Check all memory blocks in parallel to search for matching memory blocks.

3: Array Processors

Set of identical Processors synchronized to perform same instruction simultaneously on different data. They are also called SIMD.

Q5. Parallel processing advantages and challenges?

Advantages: -
1. More fault tolerance
2. Increases throughput
3. Better price/performance

Challenges:
1. Image processing
2. Large-scale simulation
3. Weather forecasting etc.

Q6. Architecture of parallel database system?

Assuming the client server architecture, parallel database system support functions can be divided into three subsystems, which are very similar as the typical RDBMS.

Depending on the architecture, processor can support all or a subset of these subsystems.

1) **Session Manager**

It acts as a transaction monitor and provides support for client-server interaction. It performs
connection and disconnection between the client process and the other two subsystems. It starts and closes the user session. In the case of an OLTP session, the session manager can trigger the execution of the preloaded transaction code with the data manager module.

2) **Request Manager**

It receives client requests for query compilation and execution. It has direct access to the database, which contains all the meta information about the data and the program. The directory itself should be managed as a database in the server. Upon request, it activates various compilation phases, triggers query execution and returns the results and error codes to the client application.

3) **Data Manager**

It provides all the low level functions needed to run compiled queries in parallel, i.e.
- Execution of DB operations
- Transaction management support - Cache management
Q7. **Parallel Join algorithms?**

Three basic parallel join algorithms for partitioned databases

---

**Parallel Nested Join (PNJ)**

This is the simplest algorithm. It composes the Cartesian product of relations R and S in parallel as shown in figure 1. Arbitrary complex join predicates may be supported.

In the **first phase**, each fragment of R is replicated at each S-node. With broadcast capability it may take m messages time, otherwise m*n.

In the **second phase**, each S-node locally joins R with the fragment Sj. This phase is done in parallel by n nodes. The local join can be done as in centralized DBMS. Depending on the local join algorithm, join processing may or may not start as soon as data are received.
To summarize, the parallel nested loop algorithm can be viewed as replacing the operator $R \bowtie S$ by

$$\bigcup_{i=1}^{n} (R \bowtie S_i)$$

**Parallel Associative Join (PAJ)**
Parallel associative join algorithm applies only for equi-join with one relation partitioned on join attribute as shown in figure 2. We assume that the equi-join predicate is on attribute $A$ from $R$ and $B$ from $S$. Relation $S$ is partitioned according to the hash function $h$ applied to join attribute $B$, meaning that all the tuples of $S$ that have same value for $h(B)$ are placed at the same node. No knowledge of how $R$ is partitioned is assumed. The algorithm precedes two phases:

In the **first phase**, relation $R$ is sent to $S$-nodes applying hash function on attribute $A$. This phase is done in parallel by $m$ nodes where $R_i$'s exist. The tuples of $R$ get distributed but not replicated across the $S$ nodes.

In the **second phase**, each $S$-node $j$ receives in parallel the relevant subset of $R(i.e., R_j)$ and joins it locally with the fragments $S_j$. Local join processing can be done in parallel. To summarize, the parallel associative join algorithm replaces the operator $R \bowtie S$ by

$$\bigcup_{i=1}^{n} (R_i \bowtie S_i)$$

**Parallel Hash Join (PHJ)**
Parallel hash join algorithm can be viewed as a generalization of parallel associative join algorithm as shown in figure 3. It also applies on equi-join but does not require any particular partitioning of operand relations. The basic idea is to partition relations $R$ and $S$ into same number $p$ of mutually exclusive sets *fragments* $R_1, R_2, \ldots, R_p$ and $S_1, S_2, \ldots, S_p$, such that

$$(R \bowtie S) = \bigcup_{i=1}^{P} (R_i \bowtie S_i)$$

The partitioning of $R$ and $S$ can be based on same hash function applied to join attribute. Each individual join is done in parallel and the join result is produced at $p$ nodes. These $p$ nodes may actually be selected at run-time based on load of the system. The main difference with the parallel associative join algorithm is that partitioning of $S$ is necessary and the result is produced at $p$ nodes rather than at $n$ $S$-nodes.
Q8. Difference b/w conservative and multi-version timestamp?

**Timestamp Ordering (TO)**
- Basic TO
- Multiversion TO
- Conservative TO

**Conservative TO**
- Basic TO generate too many restarts
- Like, if a site is relatively calm, then its transactions will be restarted again and again
- Synchronizing timestamps may be very costly
- System clocks can used if they are at comparable speeds
- In con-TO, operations are not executed immediately, but they are buffered
- Scheduler maintains queue for each TM.
- Operations from a TM are placed in relevant queue, ordered and executed later
- Reduces but does not eliminate restarts

**Multiversion TO**
- Another attempt to reduce the restarts
- Multiple versions of data items with largest r/w stamps are maintained.
- Read operation is performed from appropriate version
- Write is rejected if any older has read or written a data item

Q9..Advantages and disadvantages of shared memory?

**Shared-Memory**
Any processor has access to any memory module or disk unit through a fast interconnect as shown in figure. Examples of shared memory parallel database systems include XPRS, DBS3 and Volcano.
Advantages:
The two main advantages are:
- Simplicity
- Load balancing is excellent since can be dynamic

Drawbacks:
The three main disadvantages are:
- Cost of high interconnect
- Low extensibility
- Low availability

Hierarchical Architecture
Hierarchical architecture (also called cluster architecture), is a combination of shared-noting and shared-memory. The idea is to build a shared-nothing machine whose nodes are shared-memory.

Advantage:
The advantage of hierarchical architecture is:
- Combines the benefits of both shared memory and shared-nothing

Q10. Advantage and disadvantages of shared nothing?

Shared-Nothing:
In shares-nothing approach each processor has exclusive access to its main memory and disk unit(s) as shown in figure 2. Then each node can be viewed as a local site (with its own database and software) in a distributed database system. Therefore, most solutions designed for distributed database such as database fragmentation, distributed transaction management and distributed query processing may be reused.

Examples:
Examples of shared-nothing parallel database systems include the Teradata’s DBC and Tandem’s NonStopSQL products as well as number of prototypes such as BUBBA, EDS, GAMMA and so on.

Advantages:
Shared-nothing has following advantages:
– Cost
– Availability and extensibility

Drawbacks:
Shared-nothing has following disadvantages:
– More complex than shared memory
– Difficult load balancing

Q11. Termination conditions of Transactions?

If transaction can complete its task successfully, we say that the transaction commits. If a transaction stops without completing its tasks, we say that it aborts when a transaction is aborted its execution is stopped and all of its already executed actions are undone by returning the database to the state before their execution. This is also known as rollback.

Q12. Types and Classification of Transactions?

Types of Transactions
Transactions have been classified according to a number of criteria one criterion is the duration of transaction.
Transaction may be classified as
• On-line (short-life)
• Batch (long-life)

Transactions can be classified according to their structure. Four broad categories
– Flat (or simple) transactions
– Closed nested transactions
– Open nested transactions
– Workflow models

Online transactions are characterized by very short execution/response times and by access to a relatively small portion of the database. Examples are banking and airline reservation transactions.

Batch transactions are CAD/CAM databases, statistical applications, report generation, complex queries and image processing.

Another classification is with respect to the organization of the read write actions if the transactions are restricted so that all the read actions are performed before any write action, the transaction is called a two-step transaction.

If transaction is restricted so that a data item has to be read before it can be updated (written), the corresponding class is called restricted (or read-before-write).

If a transaction is both two-step and restricted, it is called a restricted two-step transaction.
Q13. Transformation rules of rewriting queries?

Transformation Rules

- Commutativity of binary operations
  \[ R \times S \iff S \times R \]
  \[ R \bowtie S \iff S \bowtie R \]

- Associativity of binary operations
  \[ (R \times S) \times T \iff R \times (S \times T) \]
  \[ (R \bowtie S) \bowtie T \iff R \bowtie (S \bowtie T) \]

14. Replica Transactions?

Distributed 2-PL

Distributed 2PL (D2PL) expects the availability of lock managers at each site. If the database is not replicated, distributed 2PL degenerates into primary copy 2PL algorithm. If data is replicated the transaction implements the ROWA replica control protocol.

The communication between cooperating sites that execute a transaction according to the distributed 2PL protocol is depicted in figure 4.

Coordinating TM, Participating LMs, Participating DPs

![Diagram of communication structure of Distributed 2PL]

Figure 4: Communication Structure of Distributed 2PL
15. Define the consistency and reliability aspects of a Transaction?

The consistency and reliability aspects of transactions are due to following properties:

1- **Atomicity:**  
also known as “all or none” property  
- In a transaction involving two or more discrete pieces of information, either all of the pieces are committed or none is committed.

2- **Consistency:**  
refers simply to the correctness of a transaction  
- A transaction either creates a new and valid state of data, or, if any failure occurs, returns all data to its original state i.e. before the transaction was started.

3- **Isolation:**  
- A transaction cannot reveal its results to other transactions before commitment  
- A transaction in process and not yet committed must remain isolated from any other transaction

4- **Durability:**  
Durability refers to that property of transaction which ensures that once a transaction commits, its results are permanent and cannot be erased from the database.

Q16. What are the principles to ensure the consistency and reliability of transaction? Level of consistency?
Q17. Query Optimization components, Search Space, Search Strategy, Cost function?

Query Optimization

- Components: Search space, cost model, search strategy
- Search space consists of equivalent query trees
- Search strategy could be static, dynamic or randomized
- Cost model sees response and total times...

Query optimization:

Query optimization refers to the process of producing a query execution plan (QEP) which represents an execution strategy for the query. The selected plan minimizes an objective cost functions. A query optimizer, the software module that performs query optimization, is usually seen as three components:

1. Search space
2. Search strategy
3. Cost model
1) **Search Space**

The search space is the set of alternative execution plans to represent the input query. These plans are equivalent, in the sense that the same result but they differ on execution order of operations and the way these operations are implemented. Search space consists of equivalent query trees produced using transformation rules. Optimizer concentrates on join trees, since join cost is the most effective.

2) **Search Strategy**

- Most popular search strategy is Dynamic Programming
- That starts with base relations and keeps on adding relations calculating cost
- DP is almost exhaustive so produces best plan
- Too expensive with more than 5 relations
- Other option is Randomized strategy
- Do not guarantee best

3) **Cost Model:**

An optimizer’s cost model includes cost functions to predict the cost of operators, statistics, and base data and formulas to evaluate the sizes of intermediate results.

**Cost function:**

- The cost of distributed execution strategy can be expressed with respect to either the total time or the response time.
- **Total time** = CPU time + I/O time + tr time
  - In WAN, major cost is tr time
  - Initially ratios were 20:1 for tr and I/O, for LAN it is 1:1.6
- **Response time** = CPU time + I/O time + tr time
  - TCPU = time for a CPU insts
  - TI/O = a disk I/O
  - TMSG = fixed time for initiating and recv a msgs
  - TTR = transmit a data unit from one site to another
Q18. Semantic and Structural heterogeneity?

Two types of heterogeneity:

1) **Semantic Heterogeneity**: same concepts modeled differently in different schemas: **Example**: naming, aggregation, generalization, attribute class, default value, database identifier, schema isomorphism, missing values

2) **Structural Heterogeneity**: differences in the representation of corresponding schema elements

**Examples**: Data scaling and precision, attribute integrity constraint, objects included, domain.

Heterogeneities are resolved by applying the mapping on elements. We cannot ask/expect the component databases to change. Most common semantic is naming conflict that could be Synonyms and Homonyms.

**Synonyms**: different names modeling same thing, like in our example schema ENGINEER/EMP and Salary/Sal.

**Homonym**: same name modeling different things, like title here.

Q19. Schema Translation and schema integrity?

**Schema Translation**: involves translating the component schemas into same data models. It helps to

- Compare schema elements
- Merge them

Choice of Common Data Model (CDM) is important in Schema Translation. We should prefer the semantic data models, like E-R or OO. After Schema Translation we perform Schema Integration (SI). **Schema Integration**: involves merging component schemas into a common schema (the global schema). SI involves identifying corresponding schema elements from different component databases and merging. Hampered mainly be semantic heterogeneities.
Q20. Joining Techniques QO

There are two methods:
1) Nested loops
2) Merge join

1) Nested loops:
It composes the product of the two relations. For each tuple of the external relation, the tuples of the internal relation that satisfy the join predicate are retrieved one by one to form the resulting relation. An index on the join attribute is very efficient access path for internal relation. In the absence of an index, for relations of n1 and n2 pages resp. this algorithm has a cost proportional to n1*n2 which may be prohibitive if n1 & n2 are high.

2) Merge join:
If consists of merging two sorted relations on the join attribute as shown in figure 1. Indices on the join attribute may be used as access paths. If the join criterion is equally the cost of joining two relations n1 and n2 pages, resp. is proportional to n1+n2. this method is always chosen when there is an equi join, and when the relations are previously sorted.

Q21. In context of query optimizer, which search strategy is better and in which condition?

Search Strategy
• Most popular search strategy is Dynamic Programming
• That starts with base relations and keeps on adding relations calculating cost
• DP is almost exhaustive so produces best plan
• Too expensive with more than 5 relations
• Other option is Randomized strategy
• Do not guarantee best.

Q22. SSD-1 Algorithm

SDD-1 Algorithm
The query optimization algorithm of SDD-1 is derived from an earlier method called the “hill-climbing” algorithm which has the distinction of being the first distribution query
processing algorithm. In this algorithm, refinements of an initial feasible solution are recursively computed until no more cost improvements can be made. The algorithm does not use semijoins, nor does it assume data replication and fragmentation. It is devised for wide area point-to-point networks. The cost of transferring the result to the final site is ignored. This algorithm is quite general in that it can minimize an arbitrary objective function, including the total time and response time.

- The hill climbing algorithm proceeds as follows.
- 1- The input to the algorithm includes the query graph, location of relations, and relation statistics.
- 2- Do the initial local processing
- 3- Select the initial best plan (ES0)
- – Calculate cost of moving all relations to a single site
- – Plan with the least cost is ES0
- 4- Split ES0 into ES1 and ES2
- – ES1: Sending one of the relation to other site, relations joined there
- – ES2: Sending the result back to site in ES0.
- 5- Replace ES0 with ES1 and ES2 when we should have cost(ES1) + cost(local join) + cost (ES2) < cost (ES0)
- 6- Recursively apply step 3 and 4 on ES1 and ES2, until no improvement
Phases of SDD-1:
There are four phases:
1. Initialization
2. Selection of beneficial semijoins
3. Assembly site selection
4. Postoptimization

The initialization phase generates a set of beneficial semijoins: BS= {SJ₁, SJ₂, ..., SJₖ} and an execution strategy that includes only local processing. The next phase selects the beneficial semijoins from BS by iteratively choosing the most beneficial semijoin, SJᵢ and modeling the database statistics and BS accordingly. The next phase computes the assembly site by evaluating, for each candidate site, the cost of transferring to it all the required data and taking the one with the least cost. A postoptimization phase permits the removal from the execution strategy of those semijoins that affect only relations stored at assembly site.

Q23. What are the heuristics of reducing a set of query tree? Data Localization

Input is a query tree.
Localizes the query data using the data distribution information from global schema: two steps.
2.1 Simplification: replaces the relation names with the fragments names
2.2 Fragment query is simplified and restructured to produce a good one-

Q24. Object server architecture and server-side function?
Client request objects from the server, which retrieves (recovers) them from the database and returns them to the requesting client. These systems are called object servers. In object servers, the server undertakes most of the DBMS services, with the client providing basically an execution environment for the application as well as some level of object management functionality.

Feature of the Object Server Architecture
• Objects move between clients and servers
• Object Manager helps to execute methods on both sides

Server side performs following functions:-
➢ OID implementation; it is tricky to manage OID in RAM and Disk
➢ Object Caching
➢ Object shipment between clients and servers can be optimized depending on type of clustering
➢ Lock management needed when same object being accessed (read, modified) at different places
➢ Persistent placement of objects
Feature of Page Server Architecture

• Data moves in the form of pages rather than objects
• Object management mainly performed at the client side
• Data management at the server side

Comparison

• One can’t be declared better absolutely
• If object access pattern resembles storage pattern, then page server is better
• Object server provides better concurrency as object based locking lets multiple client access objects from the same page
• Object server ships only the required objects, so data transferred is reduced

Q25. Deadlock Management?

Deadlock Management

• Locking based concurrency control generates deadlock
• T1 waits for data item being held by T2, and other way round as shown in figure 8.
• A tool in analyzing deadlocks is a Wait-for Graph (WFG).

A WFG represents the relationship between transactions waiting for each other to release data it

There are three methods for handling deadlocks:

• Prevention
• Avoidance
• Detection and resolution

Q26. Dirty read and fuzzy read with example?

Dirty Read: A transaction reads the written value of another transaction before its commitment, like,
--. W1(x), ----, R2(x), ---, C1(or A1)----, C2(or A2).

Non-repeatable or Fuzzy Read: Two reads of same data item by same transaction and a write by another transaction on the same data item
--. R1(x), ----, W2(x), ---, C2------, R1(x)----

Q27. Define serial and interleaved schedule.

Serial Schedule: If all the transactions included are executed one after another. A serial schedules always leaves the database in a consistent state.

If we have three transactions, T1, T2, T3 then one serial schedule may be: T1 ^T3 ^T2
Interleaved Schedule: A schedule is in which operations from different transactions are mixed with each other in execution.

Like $S_1 = \{W_2(x), R_1(x), R_3(x), W_1(x), C_1, W_2(y), R_3(y), R_2(z), C_2, R_3(z), C_3\}$ is an interleaved schedule.

Q28. What are the Instruction / Transactions states? Different steps of the execution of instruction.

An instruction is divided into a number of distinct stages, each allocated to distinct processor. Like, steps may be
1. Fetch instruction
2. Decode it
3. Calculate effective address of operands
4. Get operands
5. Execute operations
6. Store result

For example, there are six processors. First processor gives the output to second and starts the next instruction first step. So in this way six instructions run in pipeline manner. Determination of operation involved here. If operation is same on large amount of data, so the step fetch and decode will skip and just perform other four steps so execution will be fast. This process is called vector processing.

Q29. query likhni thi given relation ki or conjuctive or disjunctive normal form me likhna tha 30. Table and predicates were given. We have to write Sql queries for fragmentation according to predicates.

Q31. Table was given. We have to find beneficial and non-beneficial semi joins.

Q32. Lets there are Cust(ID, name, contact, location) table. This table horizontally fragmented to three sites like Cust1, Cust2, and Cust3. Apply query to get customer those location=’Islamabad’? make the tree graph of this query?

Q33. Write sql query as per conditions and give CNF and DNF forms?
Operators’ Complexity:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select, Project (without duplicate elimination)</td>
<td>O(n)</td>
</tr>
<tr>
<td>• Project (with duplicate elimination), Group</td>
<td>O(nlog n)</td>
</tr>
<tr>
<td>• Join, Semi-Join, Division, Set Operators</td>
<td>O(nlog n)</td>
</tr>
<tr>
<td>• Cartesian Product</td>
<td>O(n^2)</td>
</tr>
</tbody>
</table>

Formulas for estimating the cardinalities:

The following are the formulas for estimating the cardinalities of the result of the basic relational algebra operations

**Selection Operation:**

- Card(σF(R)) = SFS(F) * card(R)
- SFS(A = value) = 1/card(πA(R))
- SFS(A > value) = max(A) − value/(max(A) − min(A))
- SFS(A < value) = value − min(A)/((max(A) − min(A))
- SFS(p(Ai) ∨ p(Aj)) = SFS(p(Ai)) + SFS(p(Aj)) − SFS(p(Ai)) * SFS(p(Aj)).

**Cardinality of Projection:**

- Hard to determine precisely
- Two cases when it is trivial
1- When a single attribute A,
\[ \text{card}(\pi_A(R)) = \text{card}(A) \]
2- When PK is included
\[ \text{card}(\pi_A(R)) = \text{card}(R) \]

**Cartesian Product:**
- \[ \text{card}(R \times S) = \text{card}(R) \times \text{card}(S) \]

**Cardinality of Join:**
- No general way to test without additional information
- In case of PK/FK combination
  \[ \text{Card}(R \bowtie S) = \text{card}(S) \]

**Semi Join:**
- \[ \text{SFSJ}(R \bowtie AS) = \frac{\text{card}(\pi_A(S))}{\text{card}(\text{dom}[A])} \]
- \[ \text{card}(R \bowtie AS) = \text{SFSJ}(S,A) \times \text{card}(R) \]

**Union:**
- Hard to estimate
- Limits possible which are \text{card}(R) + \text{card}(S) and \( \max\{\text{card}(R) + \text{card}(S)\} \)

**Difference:**
- Like Union, \text{card}(R) for (R-S), and 0
1. How to verify correctness of vertical fragmentation?

ANS:

**Completeness**

The set of attributes \( A \) over which the relation \( R \) is defined consists of:

\[
A = U Ri
\]

Completeness of vertical fragmentation is ensured.

**Reconstruction**

The reconstruction of the original global relation is made possible by the join operation. For a relation \( R \) with vertical fragmentation \( FR \) and key attribute(s) \( K \),

\[
R = \bowtie K Ri, \forall Ri \in FR
\]

As long as each \( Ri \) is complete, the join operation will properly reconstruct \( R \). Each \( Ri \) should contain the key attribute(s) of \( R \), or it should contain the system assigned tuple IDs (TIDs).

**Disjointness**

The disjointness of fragments is not as important in vertical fragmentation as it is horizontal fragmentation.

2. what are the two methods of joining relations?

Ans:

There are two methods which is used for joining the relations:

1) nested loop
2) merge join

Nested loop: It composes the product of the two relations. For each tuple of the external relation, the tuples of the internal relation that satisfy the join predicate are retrieved one by one to form the resulting relation. In the absence of an index, for relations of \( n1 \) and \( n2 \) pages resp.

**Merge join:** If consists of merging two sorted relations on the join attribute. Indices on the join attribute may be used as access paths. If the join criterion is equally the cost of joining two relations \( n1 \) and \( n2 \) pages, resp. is proportional to \( n1+n2 \).

Example:

![Join graph of query](image-url)
3. Objective of Query optimization and write the time complexity of select, join, Cartesian product, divide, union and semi-join

The core objective of query optimization is to find most optimal result and minimize the time cost that is occurred by query processing operations.

**Time complexity:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>Time Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select, project (with duplicate)</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Project (without duplicate eliminate), group</td>
<td>(O(n \log n))</td>
</tr>
<tr>
<td>Join, semi-join, divide, union</td>
<td>(O(n \log n))</td>
</tr>
<tr>
<td>Cartesian product</td>
<td>(O(n^2))</td>
</tr>
</tbody>
</table>

4. A table was given and we perform vertical fragmentation.

**Ans:**

<table>
<thead>
<tr>
<th>Eld</th>
<th>eName</th>
<th>eDept</th>
<th>eQual</th>
<th>eSal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zafar</td>
<td>Computer</td>
<td>Master</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>Adnan</td>
<td>Commerce</td>
<td>Master</td>
<td>25000</td>
</tr>
<tr>
<td>3</td>
<td>Ali</td>
<td>Computer</td>
<td>M.Phil</td>
<td>35000</td>
</tr>
<tr>
<td>4</td>
<td>Kamran</td>
<td>Computer</td>
<td>M.Phil</td>
<td>35000</td>
</tr>
<tr>
<td>5</td>
<td>Umair</td>
<td>Commerce</td>
<td>P.hd</td>
<td>50000</td>
</tr>
</tbody>
</table>

**Vertical fragmentation**

**SELECT eId, eName, eDept FROM EMP**

<table>
<thead>
<tr>
<th>Eld</th>
<th>eName</th>
<th>eDept</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Zafar</td>
<td>Computer</td>
</tr>
<tr>
<td>2</td>
<td>Adnan</td>
<td>Commerce</td>
</tr>
<tr>
<td>3</td>
<td>Ali</td>
<td>Computer</td>
</tr>
<tr>
<td>4</td>
<td>Kamran</td>
<td>Computer</td>
</tr>
<tr>
<td>5</td>
<td>Umair</td>
<td>Commerce</td>
</tr>
</tbody>
</table>

**SELECT eId, eQual, eSal FROM EMP**

<table>
<thead>
<tr>
<th>Eld</th>
<th>eQual</th>
<th>eSal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Master</td>
<td>25000</td>
</tr>
<tr>
<td>2</td>
<td>Master</td>
<td>25000</td>
</tr>
<tr>
<td>3</td>
<td>M.Phil</td>
<td>35000</td>
</tr>
<tr>
<td>4</td>
<td>M.Phil</td>
<td>35000</td>
</tr>
<tr>
<td>5</td>
<td>P.hd</td>
<td>50000</td>
</tr>
</tbody>
</table>
5. Difference between Schema translation and schema integration, and the difference between semantic and structural heterogeneity.

**Schema Translation:**
involves translating the component schemas into same data models. It helps to Compare schema elements and Merge them. Choice of Common Data Model (CDM) is important in Schema Translation. We should prefer the semantic data models, like E-R or OO.

**Schema Integration (mixing):**

involves merging component schemas into a common schema (the global schema). SI involves identifying corresponding schema elements from different component databases and merging.

1) **Semantic Heterogeneity:**

same concepts modeled differently in different schemas:

**Example:** naming, aggregation, generalization, attribute class, default value, database identifier, schema isomorphism, missing values

2) **Structural Heterogeneity:**

differences in the representation of corresponding schema elements

**Examples:** Data scaling and precision, attribute integrity constraint, objects included, domain

6. Create Emp table queries on different sites and select and view on one site.

**Ans:**

Create view custG as select * from custPesh
Union All select * from QTA.bank.dbo.custQTA
Create view custG as select * from custQta Union
All select * from PESH.bank.dbo.custPesh
showing the global view by perform the following sql command:-
select * from custG
7. **Conjunctive and disjunctive Normal Form query from a given table.**

Ans:

Conjunctive NF: 
\[(p_{11} \land p_{12} \land \ldots \land p_{1n}) \land \ldots \land (p_{m1} \land p_{m2} \land \ldots \land p_{mn})\]

Disjunctive NF: 
\[(p_{11} \land p_{12} \land \ldots \land p_{1n}) \land \ldots \land (p_{m1} \land p_{m2} \land \ldots \land p_{mn})\]

Example:
The query expressed in SQL is:
```
SELECT ENAME
FROM EMP, ASG
WHERE EMP.ENO=ASG.ENO
AND ASG.PNO='P1'
AND DUR=12
OR DUR=24
```

The qualification in conjunctive NF is
\[\text{EMP.ENO} = \text{ASG.ENO} \land \text{ASG.PNO}='P1' \land (\text{DUR}=12 \land \text{DUR}=24)\]

The qualification in disjunctive NF is
\[((\text{EMP.ENO} = \text{ASG.ENO} \land \text{ASG.PNO}='P1' \land \text{DUR}=12) \land\]
\[\text{(EMP.ENO} = \text{ASG.ENO} \land \text{ASG.PNO}='P1' \land \text{DUR}=24)\]

Question No. 8 (Lecture 27) Write serial schedule and interleaved schedule for these transactions.

<table>
<thead>
<tr>
<th>T1:</th>
<th>T2:</th>
<th>T3:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Read(x)</td>
<td>2. Write(x)</td>
<td>3. Read(x)</td>
</tr>
<tr>
<td>4. Write(x)</td>
<td>1. Write(x)</td>
<td>6. Write(y)</td>
</tr>
<tr>
<td>5. Commit</td>
<td>8. Read(z)</td>
<td>7. Read(y)</td>
</tr>
<tr>
<td></td>
<td>9. Commit</td>
<td>10. Read(z)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11. Commit</td>
</tr>
</tbody>
</table>

Answer

Serial Schedule

T1 T3 T2

Interleaved Schedule

S1 = {W2(x), R1(x), R3(x), W1(x), C1, W2(y), R3(y), R2(z), C2, R3(z), C3}

Question No. 9 (Lecture 30) Considering the tables

- EMP(eNo, eName, title)
- ASG(eNo, pNo, resp, dur)
- PROJ(pNo, pName, budget, loc)

Query: Get the names of employees who are managing a project and normalize the query

Answer

Names of employees who are managing a project

SELECT eName
FROM EMP, ASG
WHERE EMP.eNo = ASG.eNo
AND resp = ‘Manager’

<table>
<thead>
<tr>
<th>EMP Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNo</td>
</tr>
<tr>
<td>------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PROJ Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>pNo</td>
</tr>
<tr>
<td>------</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ASG Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>eNo</td>
</tr>
<tr>
<td>------</td>
</tr>
</tbody>
</table>

Question 10 (Lecture 31) Query was given and we have to draw query graph.

Answer

Query Graph

This graph is used for most queries involving select, project, and join operations. In a graph, one node represents the result relation and any other node represents an operand.
relation. An edge between two nodes that are not results represents a join, whereas an edge whose destination node is the result represents a project.

Example
Consider the following query in SQL:
```
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](image)

**Question 11 (Lecture 33)** Ways to transfer data between different sites. Cost function: **Answer**

- The cost of distributed execution strategy can be expressed with respect to either the total time or the response time.
- Total time = CPU time + I/O time + tr time
- In WAN, major cost is tr time
- Initially ratios were 20:1 for tr and I/O, for LAN it is 1:1.6
- Response time = CPU time + I/O time + tr time
- TCPU = time for a CPU insts
- TI/O = a disk I/O
- TMSG = fixed time for initiating and recv a msgs
- TTR = transmit a data unit from one site to another

Example:
Assume that TMSG and TTR are expressed in time units. The total cost of transferring $x$ data units from site 1 to site 3 as shown in figure 4 and $y$ data units from site 2 to site 3 is

- Total Time = $2 \text{TMSG} + \text{TTR} \cdot (x+y)$
- Response Time = max\{TMSG + TTR\cdot X, TMSG + TTR\cdot Y\}

**Question 12 (Lecture 39)** Advantages and disadvantages of Shared Disk System?

**Answer**

**Shared-Disk:**

In the shared-disk approach any processor has access to any disk unit through the interconnect but exclusive (non-shared) access to its main memory as shown in figure 1. Then, each processor can access database pages on the shared disk and copy them into its own cache. To avoid conflicting accesses to the same pages, global locking and protocols for the maintenance of cache coherency are needed.

**Example:**

Examples of shared-disk parallel database systems include IBM’s IMS/VS data sharing product and DEC’s VAX DBMS and Rdb products.

**Advantages:**

Shared disk has number of advantages:
- Cost, Extensibility
- Easy migration from Uniprocessor systems
- Load balancing

**Demerits:**

Shared disk suffers from: high compexity
Potential performance problems

**Question 13 (Lecture 39)** Define two classes of NUMA architecture

**Answer**

Two classes of NUMA

- Cache coherent-NUMA: divides memory statically among all nodes
- Cache only Memory Architecture: converts per-node memory into a large cache of shared address space as shown in figure 3.

Because shared-memory and cache coherency are supported by hardware, remote memory access is very efficient, only several times (typically 4 times) the cost of local access as shown in figure. The strong argument for NUMA is that it does not require any rewriting of application software.

**Question 14 (Lecture 39)** NUMA architecture.

**Answer**

NUMA Architectures

- Non-uniform memory access
- To provide benefits of shared-memory model in a scalable parallel architecture

**Question 15 (Lecture 35)** Compare ship-whole and fetch as needed

**Answer**

1) Ship-whole
   - Entire relation transferred
   - Stored in a temporary relation
   - In case of merge-join approach, tuples can be processed as they arrive

2) Fetch-as-needed
   - External relation is sequentially scanned
   - Join attribute value is sent to other relation
   - Relevant tuples scanned at other site and sent to first site

Inter-site transfers: comparison

1. Ship-whole
   - larger data transfer
   - smaller number of messages
   - better if relations are small

2. Fetch-as-needed
   - number of messages = O(cardinality of external relation)
   - data transfer per message is minimal
   - better if relations are large and the join selectivity is good.

**Question 16 (Lecture 35)** How R* algorithm performs distributed query processing?

**Answer**

R* Algorithm:
R* uses a compilation approach where an exhaustive search of all alternative strategies is performed in order to choose the one with the least cost. Predicting and enumerating these strategies is costly, the overhead of exhaustive search is rapidly amortized if the query is executed frequently. The R* query processing algorithm deals only with relations as basic units. Query compilation is distributed task in R* coordinated by a master site, where the query is initiated. The optimizer of the master site makes all inter site decisions, such as the selection of the execution sites and the fragments as well as the method for transferring data. As in the centralized case, the optimizer must select the join ordering, the join algorithm (nested loop or merge loop), and the access path for each fragment (e.g. clustered index, sequential scan e.t.c). These decisions are based on statistics and formulas used to estimate the size of intermediate results and access path information.

The optimizer must select the sites of join results and the method of transferring data between sites. To join two relations, there are three candidate sites: the site of a first relation, the site of second relation or a third site. In R*, two methods are supported for inter-site data transfers.

1) Ship-whole
   • Entire relation transferred
   • Stored in a temporary relation
   • In case of merge-join approach, tuples can be processed as they arrive
2) Fetch-as-needed
   • External relation is sequentially scanned
   • Join attribute value is sent to other relation
   • Relevant tuples scanned at other site and sent to first site
**Question 17 (Lecture 31) Answer**

**Operator tree**

It is a tree in which a leaf node is a relation and a nonleaf node is an intermediate relation produced by a relational algebra operator. The transformation of a tuple relational calculus query into an operator tree can easily be achieved as follows. First, a different leaf is created for each different tuple variable. In SQL, the leaves are immediately available in the FROM clause. Second, the root node is created as a project operation and these are found in SELECT clause. Third, the SQL WHERE clause is translated into the sequence of relational operations (select, join, union, etc.).

**Example**

Consider the SQL query:

```sql
SELECT ENAME
FROM PROJ, ASG, EMP
WHERE ASG.ENO = EMP.ENO
AND ASG.PNO = PROJ.PNO
AND ENAME = "J.DOE"
AND PROJ.PNAME = "CAD/CAM"
AND (DUR = 12 OR DUR = 24)
```
By applying transformation rules many different trees may be found.

**Transformation Rules**

- Commutativity of binary operations
  
  \[ R \times S \leftrightarrow S \times R \]
  \[ R \bowtie S \leftrightarrow S \bowtie R \]

- Associativity of binary operations
  
  \[ (R \times S) \times T \leftrightarrow R \times (S \times T) \]
  \[ (R \bowtie S) \bowtie T \leftrightarrow R \bowtie (S \bowtie T) \]

There are other rules that we will discuss in next lecture.
Question 1: Under the context of query optimizer, which search strategy is better and in which condition? Marks-5

**Dynamic Programming:** Most popular search strategy used by query optimizers is dynamic programming, which is deterministic. Deterministic strategies processed by building plans, starting from base relations, joining one more relation at each step until complete plans are obtained.

Dynamic programming is almost exhaustive and assumes that the “best” of all plans is found.

**Randomized Strategy:** such as iterative improvement and sim-solutions around some particular points. They do not guarantee that the best solution is obtained but avoid the high cost of optimization, in term of memory and time consumption.

Question 2: Differentiate b/w hash partitioning and range portioning with the help of diagram.
Marks-15

**Round Robin:**
With n partitions, the ith tuple is placed in partition (I mod n). this strategy enables sequential access to a relation to be done in parallel. The direct access to individual tuples. Base on predicate, requires accessing the entire relation.

**Hash partitioning:** It applies a hash function to some attribute which yields the partition number as shown in figure. This strategy allows exact-match queries on the selection attribute to be processed by exactly one node and all other queries to be processed by all the nodes in parallel.

**Range Partitioning:** It distributes tuples based on value intervals of some attribute as shown in figure 6. It is suitable for exact match and range queries. Range partition can result in high variation in partition size.

Question 3: Differentiate between page server and object server Marks-5

**Objective Server:** Client request objects from the server, which retrieves them from the database and returns them to the requesting client. These systems are called object servers as shown in figure 1. In object servers, the server undertakes most of the DBMS services, with the client providing basically an execution environment for the application as well as some level of object management functionality. The object management layer is duplicated at both
the client and the server in order to allow both to perform object functions. Object manager serves a number of functions.

**Feature of the Object Server Architecture**
- Objects move between clients and servers
- Object Manager helps to execute methods on both sides

**Server Side Functions**
Server side performs following functions
- OID implementation; it is tricky to manage OID in RAM and Disk
- Object Caching
- Object shipment between clients and servers can be optimized depending on type of clustering
- Lock management needed when same object being accessed (read, modified) at different places

**Page Server:** The unit of transfer between the servers and the clients is a physical unit of data, such as a page or segment, rather than an object.

**Feature of Page Server Architecture**
- Data moves in the form of pages rather than objects
- Object management mainly performed at the client side
- Data management at the server side

**Comparison**
- One can't be declared better absolutely
- If object access pattern resembles storage pattern, then page server is better
- Object server provides better concurrency as object based locking lets multiple client access objects from the same page
- Object server ships only the required objects, so data transferred is reduced

**Question 4: Write Advantages of Replication? Marks-5**
**Question 5:** How the parallel associative join algorithm performs parallel join of a partitioned database? 
**Marks:** 10

**Parallel Associative Join:** Parallel associative join algorithm applies only for equi-join with one relation partitioned on join attribute. We assume that the equi-join predicate is on attribute A from R and B from S. relation S is partitioned according to the hash function h applied to join attribute B, meaning that all the tuples of S that have same value for h(B) are placed at the same node. No knowledge of how R is partitioned is assumed. The algorithm precedes two phases:

In the first phase: relation R is sent to S-nodes applying hash function on attribute A. this phase is done in parallel by m nodes where Ri’s exist. The tuples of R get distributed but not replicated across the S nodes.

In the second phase, each S-node j receives in parallel the relevant subset of R(i.e, Rj) and joins it locally with the fragments Sj. Local join processing can be done in parallel.

**Question 6:** Write down advantages as well as challenges of parallel processing? **Marks:** 5

**Question 7:** Define Consistency which is of the principle of transaction. Describe its levels?  
**Marks:** 10

**Consistency:** The consistency of a transaction is simply its correctness. In other words, a transaction is correct programs that map one consistent database state to another. Verifying that transactions are consistent is the concern of semantic data control on the other hand, is the objective of concurrency control mechanisms.

There is an interesting classification of consistency that parallels our discussion above and is equally important. This classification groups databases into four levels of consistency. In the following definition (which is taken verbatim from the original paper). Dirty data refers to data values that have been updated by a transaction prior to its commitment. Then, based on the concept of dirty data, the four levels are defined as follows:
Degree 3: Transaction T sees degree 3 consistency if:
1. T does not overwrite dirty data of other transactions.
2. T does not commit any writes until it completes all its writes or EOT.
3. T does not read dirty data from other transactions.
4. Other transactions do not dirty any data read by T before T complete.

Degree 2: Transaction T sees degree 2 consistency if:
1. T does not overwrite dirty data of other transactions.
2. T does not commit any writes before EOT.
3. T does not read dirty data from other transactions.

Degree 1: Transaction T sees degree 1 consistency if:
1. T does not overwrite dirty data of other transactions.
2. T does not commit any writes before EOT.

Degree 0: Transaction T sees degree 0 consistency if:
1. T does not overwrite dirty data of other transactions.

Question: Define sdd-1 algorithm.

The query optimization algorithm of SDD-1 is derived from an earlier method called the “Hill-Climbing” algorithm, which has the distinction of being the first distributed query processing algorithm. In this algorithm, refinements of an initial feasible solution are recursively computed until no more cost improvements can be made. The algorithm does not use semijoins, nor does it assume data replication and fragmentation. The hill-climbing algorithm is in the class of greedy algorithms, which start with an initial feasible solution and iteratively improve it. The main problem is that start strategies with higher initial cost, which could nevertheless produce better overall benefits, are ignored. Furthermore, the algorithm may get stuck at a local minimum cost solution and fail to reach the global minimum.

Question: Compare conservative timestamp ordering and multiversion timestamp ordering

The Conservative timestamp ordering TO algorithms attempt to lower this system overhead by reducing the number of transaction restarts. Let us first present a technique that is commonly used to reduce the probability of restarts. Remember that a TO scheduler restarts a transaction if a younger conflicting transaction is already scheduled or has been executed. Note that such occurrences increase significantly if, for example, one site is comparatively inactive relative to the others and does not issue transactions for an extended period. In this case its timestamp counter indicates a value that is considerably smaller that the counters of other sites.
The basic technique that is used in conservative TO algorithms is based on the following idea: the operations of each transaction are buffered until an ordering can be established so that rejections are not possible, and they are executed in that order. We will consider one possible implementation of the conservative TO algorithm.

**Multiversion TO Algorithm:** Multiversion TO is another attempt at eliminating the restart overhead cost of transactions. Most of the work on multiversion TO has concentrated on centralized databases, so we present only a brief overview. However, we should indicate that multiversion TO algorithm would be a suitable concurrency control mechanism for DBMSs that are designed to support applications which inherently have a notation of versions of database objects. In multiversion TO, update do not modify the database; each write operation creates a new version of that data item. Each version is marked by the timestamp of the transaction that creates it. Thus the multiversion TO algorithm trades storage space for time.

**Question: Three components of query optimizer**

Three components of query optimizer is as follow:

1. Search Space
2. Cost Model
3. Search Strategy

**Search Space:** query execution plans are typically abstracted by means of operator trees, which define the order in which the operations are executed. They are enriched with additional information, such as the best algorithm chosen for each operation. For a given query, the search space can thus be defined as the set of equivalent operator trees that can be produced using transformation rules. To characterize query optimizers, it is useful to concentrate on join trees, operator trees whose operators are join or Cartesian product. This is because permutations of the join order have the most important effect on performance of relational queries.

**Search Strategy:** Most popular search strategy used by query optimizers is dynamic programming, which is deterministic. Deterministic strategies processed by building plans, starting from base relations, joining one more relation at each step until complete plans are obtained.

**Dynamic programming** is almost exhaustive and assumes that the “best” of all plans is found.

**Randomized Strategy:** such as iterative improvement and sim-solutions around some particular points. They do not guarantee that the best solution is obtained but avoid the high cost of optimization, in term of memory and time consumption.

**Cost Model:** An optimizers cost model includes cost factions to predict the cost of operations, statistics and base data and formulas to evaluate the size of intermediate results.

Cost functions: the cost of the distributed execution strategy can be expressed with respect to either the total time or the response time. The total time is the sum of all time (also referred to as cost) components, while the response time is the elapsed time from the initiation to the completion of the query. A general formula for determining the total time can be specified as follows;

\[
\text{Total time} = T_{cpu} \times \#\text{insts} + T_{i/o} \times \#\text{I/Os} + T_{msg} \times \#\text{msgs} + T_{tr} \times \#\text{bytes}
\]

**Question: Components of cost model**

Total time: \( T_{cpu} \times \#\text{insts} + T_{i/o} \times \#\text{I/Os} + T_{msg} \times \#\text{msgs} + T_{tr} \times \#\text{bytes} \)
The two first components measure the local processing time, where T_cpu is the time of a CPU instruction and T_i/o is the time of a disk I/O. The communication time is depicted by the two last components. T_msg is the fixed time of initiating and receiving a message, while T_tr is the time it takes to transmit a data unit from one site to another. The data unit is given here in term of bytes (#bytes is the sum of the sizes of all messages).

**Query optimization in distributed DB Cartesian product:**

The cardinality of the Cartesian product of R and S is simply

$$\text{Card}(R \times S) = \text{card}(R) \times \text{card}(S)$$

**Shared-Memory:** Any processor has access to any memory module or disk unit through a fast interconnect

![Figure 6](image)

**Advantages:**
The two main advantages are:
- Simplicity
- Load balancing is excellent since it can be dynamic

**Drawbacks:**
The three main disadvantages are:
- Cost of high interconnect
- Low extensibility
- Low availability

**Question: Advantage/Disadvantage of shard disk**

Shared-Disk: In the shared-disk approach any processor has access to any disk unit through the interconnect but exclusive (non-shared) access to its main memory as shown in figure 1. Then, each processor can access database pages on the shared disk and copy them into its own cache. To avoid conflicting accesses to the same pages, global locking and protocols for the maintenance of cache coherency are needed.

![Figure 1](image)

**Advantages:** Shared disk has number of advantages:
- Cost, Extensibility
- Easy migration from Uniprocessor systems
- Load balancing

**Demerits:** Shared disk suffers from:
– Higher complexity
– Potential performance problems

Shared-Nothing: In shares-nothing approach each processor has exclusive access to its main memory and disk unit(s) as shown in figure 2. Then each node can be viewed as a local site(with its own database and software) in a distributed database system. Therefore, most solutions designed for distributed database such as database fragmentation, distributed transaction management and distributed query processing may be reused.

![Shared-nothing architecture](image)

Advantages: Shared-nothing has following advantages:
– Cost
– Availability and extensibility

Drawbacks: Shared-nothing has following disadvantages:
– More complex than shared memory
– Difficult load balancing

Question: Transformation rules

- Commutativity of binary operations
  \[ R \times S \Leftrightarrow S \times R \]

- Associativity of binary operations
  \[ (R \times S) \times T \Leftrightarrow R \times (S \times T) \]

- Idempotence of unary operations
  \[ o \pi A'(\pi A''(R)) \Leftrightarrow \pi A'(R) \]
  \[ o \sigma p_1(A1)(\sigma p_2(A2)(R)) \Leftrightarrow \sigma p_1(A1) \land p_2(A2)(R) \]

- Commuting selection with projection
\o \pi A_1, \ldots, A_n(\sigma p(A_p)(R)) \Leftrightarrow \pi A_1, \ldots, A_n((\sigma p(A_p) \pi A_1, \ldots, A_n, A_p)(R))

- Commuting selection with binary operations \o \sigma p(A)(R \times S) \Leftrightarrow (\sigma p(A)(R)) \times S

\o \sigma p(A_i)(R(A_j,B_k)S) \Leftrightarrow (\sigma p(A_i)(R))(A_j,B_k)S

- Commuting projection with binary operations \o \Pi C(R \times S) \Leftrightarrow \Pi A'(R) \times \Pi B'(S)

\o \Pi C(R(A_j,B_k)S) \Leftrightarrow \Pi A'(R)(A_j,B_k)\Pi B'(S)

**Question: Hierarchical Architecture/NUMA**
Hierarchical architecture (also called cluster architecture), is a combination of shared noting and shared-memory. The idea is to build a shared-nothing machine whose nodes are shared-memory.

**Advantage:**
The advantage of hierarchical architecture is:
- Combines the benefits of both shared memory and shared-nothing

**NUMA Architectures**
- Non-uniform memory access
- To provide benefits of shared-memory model in a scalable parallel architecture

Two classes of NUMA

Cache coherent-NUMA: divides memory statically among all nodes
- Cache only Memory Architecture: converts per-node memory into a large cache of shared address space as shown in figure 3.
Because shared-memory and cache coherency are supported by hardware, remote memory access is very efficient, only several times (typically 4 times) the cost of local access as shown in figure. The strong argument for NUMA is that it does not require any rewriting of application software.

**Dirty read and fuzzy read with example.**

Dirty Read: A transaction reads the written value of another transaction before its commitment, like, --, W1(x), ----, R2(x), ---, C1(or A1)-----, C2(or A2).

Non-repeatable or Fuzzy Read: Two reads of same data item by same transaction and a write by another transaction on the same data item

--, R1(x), ----, W2(x), ---, C2-----, R1(x)----

**Unsolved Questions:**

1. Write serial schedule and interleaved schedule for these transactions.
2. Deadlock in locking based concurrency control, how it handled
3. Paralled nested loop algorithm for paralled join in partitioned database
4. What are the design issues that arise in object database systems?
5. Differentiate between centralized and distributed 2-phase locking in the context of concurrency control.
6. How R* algorithm performs distributed query processing?
7. How the parallel associative join algorithm performs parallel join of a partitioned database?
8. Differentiate between horizontal and vertical partitioning.
9. Nested loop join & merge join difference
10. Difference b/w optimization and pessimistic transactions
Termination conditions of Transaction
Commit
Abort / rollback

Characterization of Transaction
Read set (RS)
Write set (WS)
Base set (BS)

Conflicting Operations
Two operations Oi(x) and Oj(x) are said to be in conflict, Oi = write or Oj = write (at least one of them is write and they access the same data item).

Properties of a Transaction (ACID)
1- Atomicity: also known as “all or none” property
2- Consistency: refers simply to the correctness of a transaction ----( Degrees??) 3- Isolation: A transaction cannot reveal its results to other transactions before commitment (Isolation levels??)
4- refers to that property of transaction which ensures that once a transaction commits, its results are permanent and cannot be erased from the database

Types of Transactions
According to duration of transaction
• on-line (short-life)
• batch (long-life)

With respect to the organization of the read write actions
• Two-step (all reads before any write)
• Restricted (Read an item before write)
• Action model (Read/write on an item to be atomic

According to their structure
• Flat (or simple) transactions
• Closed nested transactions
• Open nested transactions
• Workflow models

Distributed Concurrency Control
CC concerns synchronizing concurrent transactions maintaining consistency of the database and maximizing degree of concurrency
• Schedule or History
Concurrent Control Algorithms

Different categorizations possible, like, mode of distribution, network topology. Synchronization primitive is the most common, Locking and Ordering.

1. **Pessimistic** (synchronizes transactions early)

   - Locking-based (Two-Phase Locking, strict 2-PL, Centralized 2PL, Primary Copy 2-PL, Distributed 2-PL)
     
     Sequence of phases: Read (R) → Computation (C) → Validation (V) → Write (W)

     i. Centralized Locking
     ii. Primary Copy Locking
     iii. Distributed Locking

   - Timestamp Ordering TO (Properties of timestamp: Uniqueness, Monotonically)
     
     I. Basic TO
     II. Multiversion TO
     III. Conservative TO

   - Hybrid

2. **Optimistic** (late in execution life cycle of transactions)

   Sequence of phases: Validation (V) → Read (R) → Computation (C) → Write (W)

   **Rules / Test validations:** Rule1, Rule2, Rule3.

   - Locking-based
   - Timestamp ordering-based

Deadlock Management

There are three methods for handling deadlocks:

- Prevention
- Avoidance
Detection and resolution

**Query processing**
The main function of a relational query processor is to transform a high level query into an equivalent lower level query. The low level query actually implements the execution strategy for the query. The transformation must achieve both correctness and efficiency.

- Centralized QP
- Distributed QP

**Objective of Query Processing**
The objective of query processing in a distributed context is to transform a high level query on a distributed database. An important of query processing is query optimization.

**Query Optimization**
Many execution strategies are correct transformations of the same high level query the one that optimizes (minimizes) resource consumption should be retained.

**Operators’ Complexity**
Relational algebra is the output of query processing. The complexity of relational algebra operations, which directly affects their execution time, dictates some principles useful to a query processor. Figure 2 shows the complexity of unary and binary operations.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select, Project (without duplicate elimination)</td>
<td>(O(n))</td>
</tr>
<tr>
<td>Project (with duplicate elimination), Group</td>
<td>(O(n\log n))</td>
</tr>
<tr>
<td>Join, Semi-Join, Division, Set Operators</td>
<td>(O(n\log n))</td>
</tr>
<tr>
<td>Cartesian Product</td>
<td>(O(n^2))</td>
</tr>
</tbody>
</table>

**Characterization of Query Processors**
There are some characteristics of query processors that can be used as a basis for comparison.

- Types of Optimization
  - I. Exhaustive search
  - II. Heuristics
Optimization Timing
I. Static: during compilation
II. Dynamic: during execution

- Statistics
  I. Relation/Fragment
  II. Attribute

- Decision Sites
  I. Centralized
  II. Distributed
  III. Hybrid

Query Decomposition:
Query decomposition transforms an SQL (relational calculus) query into relational algebra query on global relations. The information needed for this transformation is found in the global conceptual schema.

4 Steps in query decomposition:

- **Normalization**
  I. Conjunctive NF
  II. Disjunctive NF
  III. Equivalence
    rules:
    1. \( p_1 \land p_2 \equiv p_2 \land p_1 \)
    2. \( p_1 \lor p_2 \equiv p_2 \lor p_1 \)
    3. \( p_1 \land (p_2 \land p_3) \equiv (p_1 \land p_2) \land p_3 \)
    4. \( p_1 \lor (p_2 \lor p_3) \equiv (p_1 \lor p_2) \lor p_3 \)
    5. \( \neg (\neg p_1) \equiv p \)

- **Analysis**
  I. Type incorrect
  II. Semantically incorrect
  III. Query Graph
  IV. Join graph

- **Elimination of redundancy**
Idempotency rules:
1. \( p \land p \Leftrightarrow p \)
2. \( p \lor \text{false} \Leftrightarrow p \)
3. \( p \land \text{false} \Leftrightarrow \text{false} \)
4. \( p \lor \text{true} \Leftrightarrow \text{true} \)
5. \( p \land \neg p \Leftrightarrow \text{false} \)
6. \( p \lor \neg p \Leftrightarrow \text{true} \)
7. \( p_1 \land (p_1 \lor p_2) \Leftrightarrow p_1 \)
8. \( p_1 \lor (p_1 \land p_2) \Leftrightarrow p_1 \)

**Rewriting**

I. Straightforward transformation of query from relational calculus into relational algebra

II. Restructuring of relational algebra to improve performance (Operator tree) By applying transformation rules many different trees (Operator tree) may be found.

**Transformation Rules**

These rules enables the generation of many equivalent trees. I. Commutativity of binary operations

II. Associativity of binary operations

III. Idempotence of unary operations

IV. Commuting selection with binary operations

V. Commuting projection with binary operations

These rules can be used in four different ways:

- They allow the separation of unary operations, simplifying the query expression.
- Unary operations on the same relation may be grouped together
- Unary operations can be commuted with binary operations
- Binary operations can be ordered
**Data Localization:**
The localization layer translates an algebraic query on global relations into an algebraic query expressed on physical fragments. Localization uses information stored in the fragment schema.

**Query optimization:**
Query optimization refers to the process of producing a query execution plan (QEP) which represents an execution strategy for the query. The selected plan minimizes an objective cost functions. A query optimizer, the software module that performs query optimization.

**Three components:**
- **Search space**
  Query optimizers restrict the size of the search space they consider. Two restrictions are:
  1. Heuristics
  2. Shape of join Tree
  Two types of join trees are distinguished:
  1. Linear Tree
  2. Bushy tree

- **Search strategy**
  1. Dynamic Programming
  2. Randomized strategy

- **Cost model**
  1. Cost function
  2. Database Statistics
  3. Cardinalities of Intermediate Results
Centralized Query Optimization:

System R:
System R performs static query optimization based on the exhaustive search of the solution space.

Two major steps in Optimization Algorithm
- Best access path for individual relation with predicate
- The best join ordering is eliminated

I. Nested loops
   II. Merge join

Join Ordering in Fragmented Queries:
Ordering joins is an important aspect of centralized query optimization. Join ordering in a distributed context is even more important since joins between fragments may increase the communication time.

Two basic approaches exist to order joins in fragment queries.

- Optimize the ordering of joins
- Replaces joins by combination of semi-joins to minimize communication cost

Join Ordering:
Semijoin based Algorithms:
The join of two relations R and S over attribute A, stored at sites 1 and 2, resp. can be computed by replacing one or both operand relations by a semijoin with the other relation, using the following rules: So $R \bowtie A S$ can be replaced:

- $(R \bowtie A S) \bowtie A S$
- $R \bowtie A (S \bowtie A R)$
- $(R \bowtie A S) \bowtie A (S \bowtie A R)$

Distributed Query Processing Algorithms:
Three main representative algorithms are
- Distributed INGRES Algorithm
- R* Algorithm

In R*, two methods are supported for inter site data transfers.
  I. Ship-whole
  II. Fetch-as-needed

Inter-site transfers: comparison

Four join strategies
- SDD-1 Algorithm

The hill climbing algorithm proceeds in 6 steps??

The improved version makes extensive use of semijoins. The objective function is expressed in terms of total communication time. Finally, the algorithm uses statistics on the database, called database profiles, where a profile is associated with a relation.

4 Phases of SDD-1:
1. Initialization
2. Selection of beneficial semijoins
3. Assembly site selection
4. Postoptimization

Parallel Processing
Computer programs generally execute sequentially, reasons are:
- Single CPU
- Programming language support
- Programmer familiarity
- Logic of program is sequential
Requirements for parallel processing:
- Computer Hardware
- An operating system capable of managing multiple processors
- Application software

Why parallel processing:
Some Applications need computing power of parallel system
- image processing
- large-scale simulation
- Weather forecasting etc.

Advantages:
Parallel processing has following advantages:
- More fault tolerance
- Increases throughput
- Better price/performance
- Synchronization cost has to be kept minimum
- Administering is difficult
Many computer programs can be broken into distinct parallel procedures

1. Non-Sequential setups
Pipeline and Vector Processors
Pipeline processors permit overlapping of instructions. This is also called producer-consumer. Stages:
1. Fetch instruction
2. Decode it
3. Calculate effective address of operands
4. Get operands
5. Execute operations
6. Store result

Multicomputer Systems
Multiple computers linked through a network in a small area.
2. Parallel Execution
Multiprocessor Systems
Multiprocessing is the use of two or more central processing units (CPUs) within a single computer system.
Interconnection systems are
- Time shared or common bus
- Crossbar switches
- Multiport-memory systems
- Multistage networks

**Associative Processors**
They are designed to speed up search for data items in memory. Parallel examination of all memory block to search matching ones.

**Array Processors (SIMD)**
Set of identical Processors synchronized to perform same instruction simultaneously on different data. They are also called SIMD. That is it about Parallel Computing basics. Let’s move to now parallel databases.

**Parallel database:**
Parallel database systems combine database management and parallel processing to increase performance and availability. The major problems for DBSs have been I/O bottleneck. Initially database machines (DBMs) designers tackled this problem through special purpose hardware. The objectives of parallel database systems can be achieved by extending distributed database technology, e.g, by partitioning the database across multiple (small) disks so that much inter- and intra- query parallelism can be obtained.
  - Inter-query parallelism
  - Intra-query parallelism

A Parallel database system acts as a DB server to multiple application servers in now clientserver organization in computer networks. It supports - DB functions
- C/S interface
- Also general purpose computing

**Advantages**
A parallel database system should provide the following advantages:
  - High Performance
  - High availability
  - Extensibility I. Linear scaleup II. Linear speedup

**Architecture of a parallel database system**
**General** architecture of a parallel database system Figure (lec 38) Three subsystems
  - Session manager
  - Request manager
• Data manager

**Parallel System Architectures**
A parallel system represents a compromise in design choices in order to provide the better cost/performance. Parallel system architectures are:

- Shared Memory
- Shared Nothing
- Shared Disk
- Hierarchical/NUMA (Two classes of NUMA)

  I. Cache coherent-NUMA: divides memory statically among all nodes
  II. Cache only Memory Architecture: converts per-node memory into a large cache of shared address space

**Comparison**

- For small configurations, shared-memory can perform the best due to better load balancing
- NUMA for mid-range and hierarchical for large scale systems

**Parallel Database Techniques**
Implementation of PDBS relies on DDBS techniques. The transaction management solutions can be reused. The critical issues for such architectures are:

- **Data placement**
  There are three basic strategies for data partitioning:
  I. Round Robin
  II. Hash partitioning
  III. Range Partitioning

**Data Replication in PDBS**
- Maintain two replicas, primary and backup copies on two separate nodes
- Interleaved partitioning
- Chained partitioning

- **Query parallelism** I. Inter-query parallelism
  II. Intra-query parallelism
    - Inter-Operation
      1. Pipeline parallelism
      2. Independent parallelism
    - Intra-Operation

- **Parallel data processing**
  Three basic parallel join algorithms for partitioned databases:
  I. Parallel Nested Loop (PNL)
  II. Parallel Associative Join (PAJ)
  III. Parallel Hash Join (PHJ)
- **Parallel query optimization**
  A query optimizer consists of three components:
  I. Search Space  
  II. Cost Model  
  III. Search Strategy

**ADDITIONAL QUESTIONS:**
1. **Define Distributed Database.**
   A logically interrelated collection of shared data (and a description of this data) physically distributed over a computer network.
2. **Define Distributed DBMS.**
   The software system that permits the management of the distributed database and makes the distribution transparent to users.
3. **Define Distributed processing.**
   A centralized database that can be accessed over a computer network.
4. **Define Parallel DBMS.**
   A DBMS running across multiple processors and disks that is designed to execute operations in parallel, whenever possible, in order to improve performance.
5. **What are the main architectures for parallel DBMS?**
   - Shared Memory  
   - Shared Disk  
   - Shared nothing
6. **What is Homogeneous and Heterogeneous DDBMS?**
   In Homogeneous system, all sites use the same DBMS product. In Heterogeneous system, sites may run different DBMS products.
7. **Define Global Conceptual Schema.**
   Global Conceptual Schema (GCS) is a logical description of the whole database, as if it were not distributed. It contains definitions of entities, relationships, constraints, security and integrity information. It provides physical data independence from the distributed environment.
8. **What are the types of Fragmentation?**
   The two types of fragmentation are  
   - Horizontal fragmentation: subset of tuples.  
   - Vertical fragmentation: subset of attributes.
9. **What are the objectives of definition and allocation of fragments?**
   The main objectives are:  
   - Locality of reference.  
   - Improved reliability and availability
10. **What are the four alternative strategies regarding placement of data?** The four alternative strategies regarding placement of data are:
   - Centralized
   - Fragmented (partitioned)
   - Complete replication
   - Selective replication

11. **What are the rules that have to be followed during fragmentation?**
    The three correctness rules are:
    - Completeness: ensure no loss of data
    - Reconstruction: ensure Functional dependency preservation
    - Disjointness: ensure minimal data redundancy.

12. **What are the objectives of concurrency control?**
    - To be resistant to site and communication failure.
    - To permit parallelism to satisfy performance requirements.
    - To place few constraints on the structure of atomic actions.

13. **What is multiple-copy consistency problem?**
    It is a problem that occurs when there is more than one copy of data item in different locations and when the changes are made only in some copies not in all copies.

14. **What are the two approaches in distributed environment?**
    - **Locking**
      Locking guarantees that the concurrent execution is equivalent to some unpredictable serial execution of those transactions.
    - **Timestamp**
      Timestamp guarantees that the concurrent execution is equivalent to specific serial execution of those transactions, corresponding to the order of timestamps.

15. **What are the types of Locking protocols?**
    - Centralized 2 phase locking
    - Primary 2 phase locking
    - Distributed 2 phase locking
    - Majority locking

16. **What are the failures in distributed DBMS?**
    - The loss of message
    - The failure of a communication link
    - The failure of a site
    - Network partitioning

17. **Define coordinator and participant.**
Every global transaction has one site that acts as the coordinator (or transaction manager) for that transaction, which is generally the site at which the transaction was initiated. Sites at which the global transaction has agents are called participants (or resource managers).

18. Define unilateral abort.
If a participant votes to abort, then it is free to abort the transaction immediately; in fact any site is free to abort a transaction at any time up until it votes to commit. This type of abort is known as unilateral abort.

19. Give the states of the coordinator
The four states of the coordinator are:
- INITIAL
- WAITING
- DECIDED
- COMPLETED

Pessimistic protocols choose consistency of the database over availability and hence do not allow transactions to execute in a partition if there is no guarantee that consistency can be maintained. The protocol uses pessimistic control algorithm.

21. What is replication?
The process of generating and reproducing multiples copies of data at one or more sites is called replication.

22. What are synchronous and asynchronous replications?
In synchronous replication the replicated data is updated immediately when the source data is updated. This is done by using 2PC protocol. In asynchronous replication the target data is updated after the source database has been modified. The delay in regaining consistency may range from a few seconds to several hours or even days.

23. What is data ownership? What are the types of ownership?
Data ownership relates to which site has the privilege to update the data. The main types of ownership are master/slave, workflow, and update-anywhere.

24. What are the implementation issues in data replication?
The main implementation issues are
- Transactional updates
- Snapshots and database triggers
- Conflict detection and resolution

25. What are the mechanisms proposed for conflict resolution?
The mechanisms proposed for conflict resolution are
- Earliest and latest timestamps
- Site priority
- Additive and average updates
- Minimum and maximum values
- User defined
26. **What is generic relational algebra tree?**
The relational algebra tree formed by applying the reconstruction algorithm is known as the is generic relational algebra tree.

27. **What are the Benefits of replication?**
Replication provides a number of benefits, including improved performance, increased reliability & data availability, and support for mobile computing & data warehousing.

28. **What are the types of replication?**
- Read-only snapshots
- Updateable snapshots
- Multimaster replication
- Procedural replication

29. **Define Snapshot logs.**
A snapshot log is a table that keeps track of changes to a master table. A snapshot log can be created using the CREATE SNAPSHOT LOG statement.

30. **Define database links.**
Database links define a communication path from one Oracle database to another database.

31. **Explain Distributed Relational Database design in detail.**

32. **Objectives of data allocation and fragmentation**
- Data Allocation:
  Strategic Objectives: Centralized, Fragmented, Complete Replication, Selective Replication.
- Fragmentation:
  - Need for fragmentation
  - Correctness of fragmentation
  - Types of fragmentation

33. **Explain about Transparencies in a DDBMS.**
- Distribution transparency
- Fragmentation transparency  Transaction transparency
- Location transparency  Concurrency transparency
- Replication transparency  Failure transparency
- Local mapping transparency  Performance transparency
- Naming transparency  DBMS transparency

34. **Explain the Functions and Architecture of a DDBMS.**
Functions of DDBMS
- Extended communication services
- Extended system catalog
- Distributed Query processing
- Extended security control
- Extended concurrency control
- Extended recovery services Reference Architecture for DDBMS
35. **Component Architecture for DDBMS**
   - § Local DBMS component (LDBMS)
   - § Data Communication component (DC)
   - § Global System Catalog (GSC)
   - § Distributed DBMS component (DDBMS)

36. **Explain in detail the recovery techniques in distributed databases.**
   - Failures in a distributed environment
   - How failures affect recovery
   - Distributed recover protocols
     - Two-phase commit (2PC)
     - Termination protocols
     - Recovery protocols
     - Election protocols
     - Communication topologies for 2PC
   - Three-phase commit (3PC)
   - Network partitioning

37. **Discuss about the X/Open Distributed Transaction Processing model.**
   - X/Open established the Distributed Transaction Processing (DTP) working group with the objective of fostering appropriate programming interface for Transaction Processing.
   - X/Open interfaces with diagram
   - Transaction Manager
   - Resource Manager
   - Procedures of TX interface
   - X/Open interfaces in a distributed environment

38. **Explain about distributed query optimization. Query optimization.**
   - Techniques in query optimization
   - Distributed query transformations
   - Reconstruction Algorithms
   - Generic relational algebra tree
   - Reduction techniques for various types of fragmentation Distributed joins.
39. Types of Deadlock Detection.
   • Centralized
   • Hierarchical
   • Distributed