1605023

question – 1.1

explain the implications of storage to MMDBMS and Disk-based DBMS.

answer

MMDBMS

=> main memory is primary storage. So, its processing or operational speed is faster. But, in terms of cost per memory unit, it is costly. Real time storage application and high performance application (where cost can be compromised with) use MMDBMS.

Disk-based DBMS

=> disk is secondary storage and non-volatile unlike main memory. Its processing is slower due to fetching of data from disk to memory being time consuming. But, its cost per memory unit is relatively economic. Existing DBMS types lie in this category.

Ans.

Answer- 2.1

1605023

i. linear ⟶ speed up = $\frac{10\,ms}{5\,ms}$ = 2 for 2x node. (linear)

ii. nonlinear/sublinear ⟶ speed up = $\frac{10\,ms}{4\,ms}$ = 2.5 for 3x node.

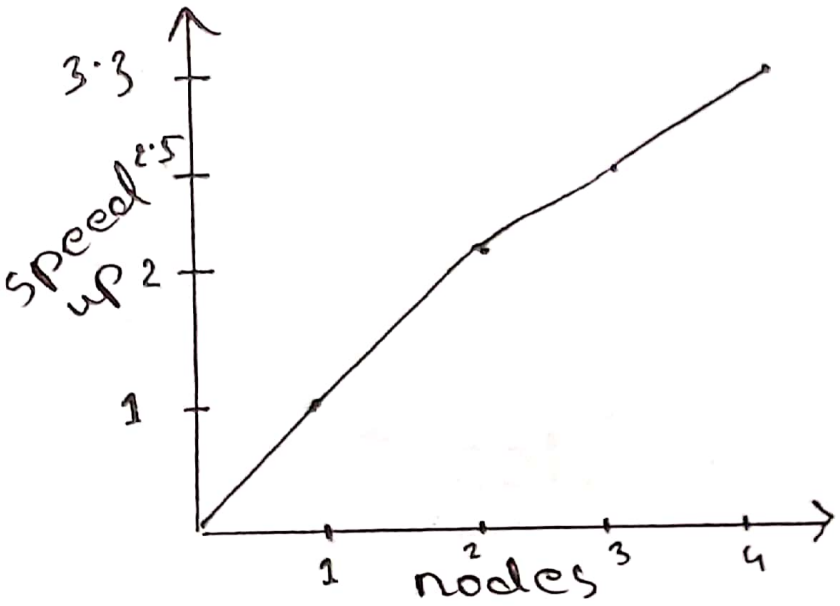
iii. nonlinear/sublinear ⟶ speed up = $\frac{10\,ms}{3\,ms}$ = 3.33 for 4x node

} (sublinear)

=) here, the speed-up is sublinear as the graph plotted against #resource is not a straight line. Ans.

graph

1605023



Ans.

answer – 2.2

1605023

i. # node = 4 and problem size = 4p, time = 40 ms

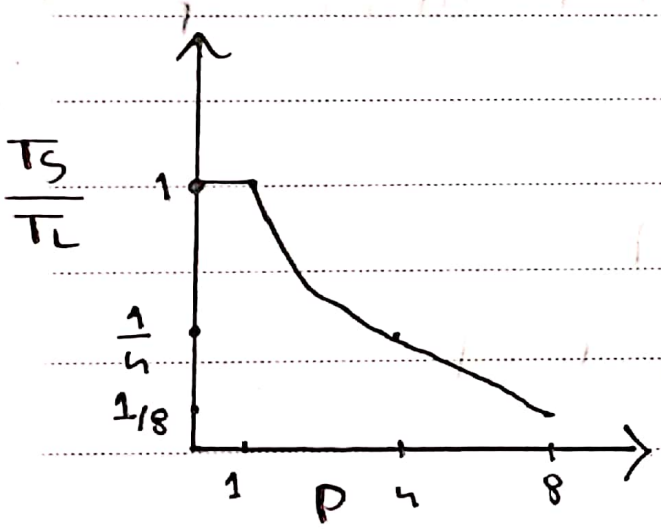
ii. # node = 8 and problem size = 8p, time = 80 ms

=> Intially, time = 10 ms for 1 node and p sized problem

Here, $\frac{T_S}{T_L} = \frac{10}{40} = \frac{1}{4}$ and $\frac{T_S}{T_L} = \frac{10}{80} = \frac{1}{8} \neq 1$.

So, scale-up graph is sublinear.

graph



- elapsed/response time increases despite increased number of nodes.

Ans.

answer - 2.3

1605023

i. $p = 1 \longrightarrow$ speed-up $= \dfrac{1}{1/n} = n$

$\Rightarrow$ that is, we observe a linear speed-up.

ii. $p = 0 \longrightarrow$ speed-up $= \dfrac{1}{1 + \frac{0}{n}} = 1$

$\Rightarrow$ elased time is unchanged as no parallelism introduced.
(constant)
no speed-up

iii. $0 < p < 1 \longrightarrow$ speed-up $= \dfrac{1}{(1-p) + \frac{p}{n}}$

$\Rightarrow$ speed-up is sublinear as the denominator decreases with increase in node number.

Ans.

- no speed-up is neither linear nor sublinear.

1605023

Date :

answer-2.4

i. $p = 1 \longrightarrow$ scale-up $= \dfrac{1}{1} = 1$

$\Rightarrow$ linear scale-up as $\dfrac{T_S}{T_L} = 1$.

ii. $p = 0 \longrightarrow$ scale-up $= \dfrac{1}{(1-0).n} = \dfrac{1}{n}$

$\Rightarrow$ sublinear scale-up as $\dfrac{T_S}{T_L} < 1$. (But, no extra node as task is done sequentially).

$\Rightarrow$ not actual scale-up because we can not leverage from introducing parallelism.

$\Rightarrow$ so, no scale-up as tasks have to be done in parallel.

Ans.

answer -3.1

1605023

Bus

⊕ → simple and less wired architecture, cheap

⊖ → may introduce increased competition for resource (interference), single point of failure(main bus)

mesh

⊕ → reduced interference due to increased connectivity

⊖ → increased hop count introduces latency

⇒ better than Bus.

Hypercube

⊕ → in terms of scalability and performance, better.

⊖ → still each node has $\log_2 n$ neighbor so latency exists.

⇒ better than first two.
⇒ expensive

tree

⊕ → Scalabity is, that is. scaleup is maintained (near linear)

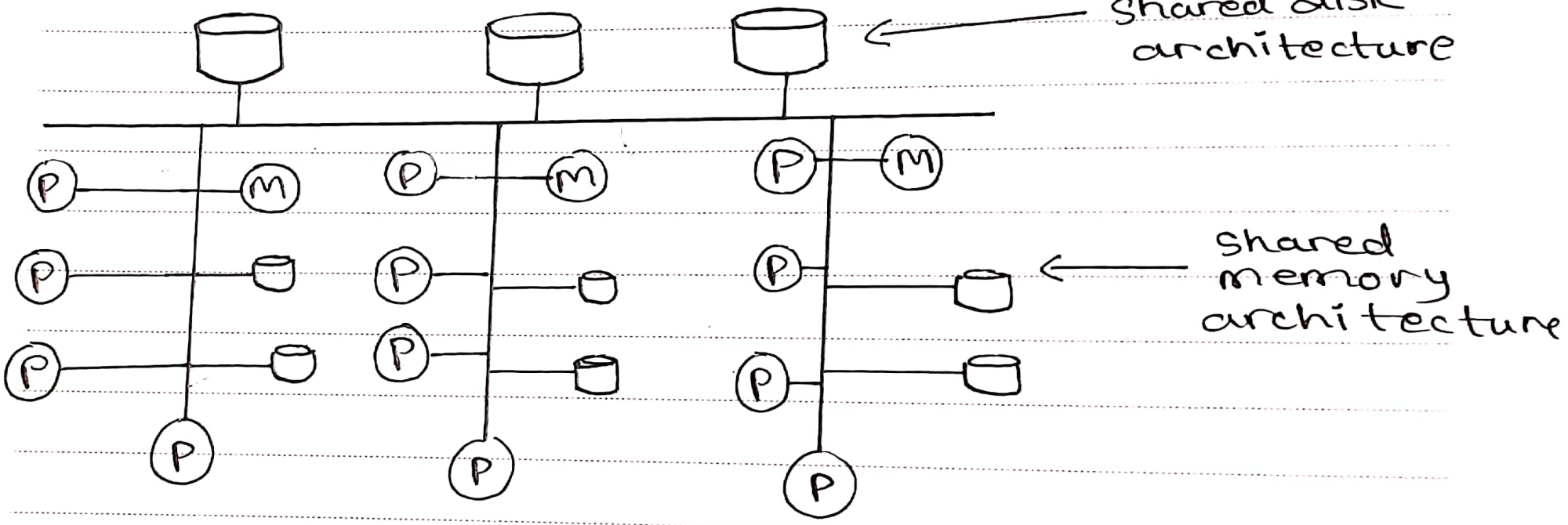
⊖ → ineresed cost, expensive.

⇒ better performance

Ans.

answer – 3·2

1605623



Ans.

answer–4.1

1605023

- Round-robin technique for tuples distribution in nodes follows the following formula:

i-th tuple goes to (i mod n)th node where

n = total number of nodes.

so, if we have tuples t0, t1, ----, t39, t40, ----

and n = 40,

then t0 goes to (0 % 40) = 0 th node

t1 goes to (1 % 40) = 1 th node

and t39 goes to (39 % 40) = 39 th node.

Date :

tho tuple again returns to 0th node and so on.

Ans.

answer – 4·2

1605023

Here, nodes are n0, n1, ---, n39 in a data center. and tuples with attributes are t0, t1, ---, t40, ----.

So, n = 40 nodes.

i. if NID alone is used as partitioning attribute in hashing, then a hash function designed will generate a value ranging from 0 to 39 based on provided NID. That tuple goes to the hash_value-th node.

ii. using street, city, district (combined character count, location code, etc.) as partitioning attribute to do hashing, the partitioning will work similarly as case – i.

Ans.

answer-4.3

1605023

in a data center,
nodes are $n_0, ---, n_9, n_{10}, ----, n_{19}$; so, $n = 20$ nodes.
tuples are $t_0, t_1, -----, t_{399}$.
ID ranges from 202105001 to 202105400.
So, considering last 3 varying digits.

a. a partition vector based on ID is given:

$V = [v_0, v_1, ---, v_{18}]$. where $v_0 = 21, v_1 = 41, v_2 = 61,$

$----, v_{17} = 361, v_{18} = 381.$

this partition vector is derived by grouping each partition with $\frac{400}{20} = 20$ tuples.

b. here, P0 contains ID from 1 to 20 as $v_0 = 21$.
P1 contains ID from 21 to 40 as $v_0 \leq v < v_1$
or $21 \leq v < 41$.

and P19 contains all IDs greater than 380
as $v_{18} = 381$ and $381 \leq v \leq 400$.

Ans.

answer-4.4

1605023

considering the given schema Person and Parents, we need to process the query:

SELECT * FROM person r, parents s WHERE r.age= s.f-age

a. given partitions are: P1, P2, P3, P4 for both relations. node-i contains partition-i from both Person and Parents tables. Also, query processing is done parallely in 4 nodes.
Here, both relations Person and Parents are partitioned based on age (f-age for parents), a node contains/joins the partitions for same age range from both relation.
So, above query where age equality is checked will be executed by 4 node parallely. If the query specifically asks for an age range, then that query may be executed by one or few nodes leaving other nodes available for other queries. If only one node is needed for processing the query, then the processing will be reduced to $\frac{1}{4}$th of that of a normal case.

b. speed-up: if the query is processed in more than one node, then a linear or sublinear speed up will be observed as processing / task is devided among nodes.

scale-up: new nodes need to be introduced to the system to maintain a linear/ sublinear scale-up. if problem size or query range is increased.

Ans.

answer-5.1

1605023

- Round-robin partitioning
- skewing হতে পারে
chance কম।

Here, we need to explain the performance of Round-Robin partitioning technique for the following scenario:

a. Scanning the entire relation

round-robin is best suited for sequential scan of entire relation on a query as all nodes have almost equal number of tuples.

b. point query

if we know which tuple to look for beforehand, then we can directly retrieve from a specific node.
Else, a sequential scan is required which takes time.

c. range query

In this case, the performance will not be good as now the query needs to be processed at all nodes. It will take a lot of time and no node can process other query then.

Ans.

1605023

answer – 5.2

Here, we need to explain the performance of hash partitioning technique for the following scenario: Date :

a. scanning the entire relation

Hash partitioning is good for sequential access given that hash function is good and partitioning is done on key attributes. Then, tuples are equally distributed among nodes.

b. point query

It performs well in partitioning attribute query. Single node is looked up and other nodes are available for processing other queries.

c. range query

It is inefficient when it comes to range queries because all nodes must be processing the query then. So, parallelism, here, can not be leveraged.

Ans.

answer-5.3

1605023

Here, we need to explain the performance of range partitioning technique for the following scenario:

a. scanning the entire relation

range partitioning provides data clustering by partitioning attribute value. So, it performs good in sequential scanning of tuples.

b. Point query

It performs well in point query on partitioning attribute because only one node needs to be accessed.

c. range query: for range queries, one or few nodes may need to be accessed (query on partitioning attribute). Then, other nodes are available for other queries. So, it performs well if required tuples are from one to a few blocks. Otherwise, potential parallelism in disk access is wasted. (example of execution skew) Ans.

1605023

answer - 6.1

a. Hash partitioning

- certain part. value may repeat in many tuples, so attribute-value, thus, data-distribution skew occurs.

b. range partitioning

- tuples are grouped based on a partitioning value. So, attr.-value skew, thus, data-distribution skew occurs.

a. Hash Partitioning

- partition skew is less likely to occur as partitioning is based on a hash function.
- execution skew is less likely unless a tuple with a specific value is added multiple times to same node by hashing.

b. range partitioning

- partition skew is very likely in the case of a badly chosen range-part. vector.
- execution skew may occur if tuples are partitioned based on an attribute like date. Then, all recent tuples will end up in the same node.

Ans.

answer-6.2

1605023

Date

(a.)

The given histogram is an equi-width histogram.

(b.)

definition of an appx. partition vector:

$V = [v_1, v_2, v_3]$ for nodes N0, N1, N2, N3.

$\Rightarrow V = [3, 8, 17]$.

This vector is obtained by the following calculation-

total tuple = $45 \times 5.0 + 35 \times 5.0 + 25 \times 5.0$

$+ 50 \times 5.0 + 15 \times 5.0$

$\Rightarrow$ total tuple = 850

so, each node may have approximately = $\frac{850}{4} = 212-213$ tuples

So, $v1 = 3$ and N1 contains 2 tuples $(1 \le v < 3)$

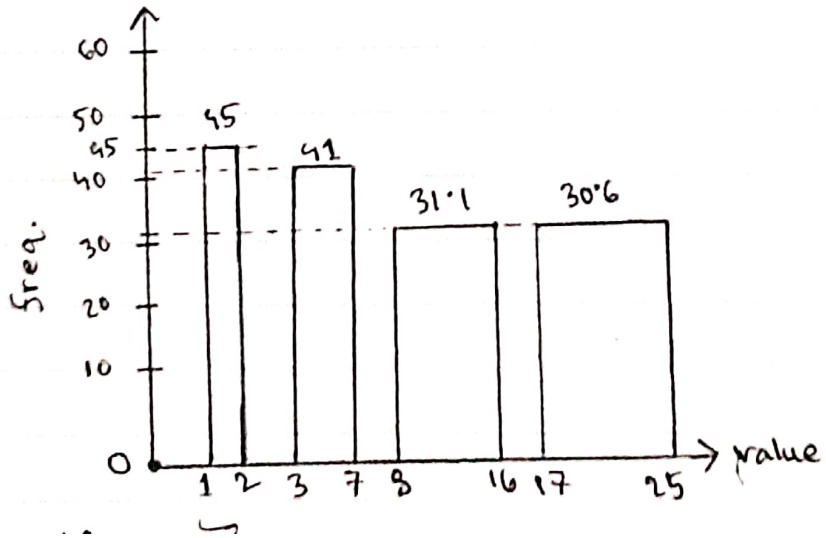
$v2 = 8$ and N2 contains 5 tuples $(3 \le v < 8)$

$v3 = 17$ and N3 contains 9 tuples $(8 \le v < 17)$

and thus, N4 contains other 9 tuples from value 17 to 25. $(17 \le v \le 25)$.

Here, uniform dist. within each range of histogram is assumed.

(c.)



Ans.

answer – 7.1

1605023

We need to explain how data distribution skew and execution skew can be handled using virtual node partitioning.

Data Distribution Skew

i. virtual node partitioning basically distributes data distribution skew among several nodes. Also, a tuple can be shifted from a highly loaded node to less loaded node with mapping table. Thus, data distribution skew is handled.

Execution Skew

i. Data execution skew, that is, one node is targeted for processing a sequence of queries in a certain period of time. To distribute this load among several nodes, tuples can be redistributed among less loaded nodes with mapping table.

ii. Node replication using virtual node partitioning can also address the issue. A specific node facing a significant number of requests in a short time span can replicate its data among several other nodes to reduce load.

Date :

Ans.

answer — 7.2

1605023

We need to explain how a query is executed in parallel storage system with dynamically partitioned storage of a relation. Here, the query and the partitioning is on the same attribute.

In a distributed/parallel data storage system with virtual node partitioning, the partitioning table is usually in the master node or it is replicated and distributed among routers and client nodes for faster query processing.

This partitioning table helps corresponding node or router to do the mapping from virtual to real node for a particular query. Thus, partitioning table diverts queries in these nodes to appropriate real node containing data. Also, dynamic repartitioning consistently changes the partitioning table. So, these updates are replicated in all the instances of the table. Thus, query diversion is achived properly.

Ans.

answer – 8.1

1605023

Advantages and disadvantages of replication are discussed below –

advantages

i. availability of data when a node fails. Other nodes containing replicated data provides what client asks for.

ii. faster operation and response.

iii. availability of real-time data.

disadvantages

i. concurrent update of data is difficult.

ii. increasing storage size.

Ans.

answer - 8.2

1605023

We are using 64MB block size to store a file named "Yourld_HFDS" of size 10GB in Hadoop File System.

=> here, block number = 10GB ÷ 64MB = 160.

=> assuming, we have 10 nodes.

Then, 160 blocks can be distributed among these 10 nodes using Round-Robin partitioning as there is no scope of skewing.

=> now, each node contains 16 blocks.

- The NameNode will contain 160 blockID entries against given file name.

=> each datanode contains 16 blocks of data and we have 10 DataNodes for storing blocks.

- also, 10 more DataNodes will contain the replica of these 160 blocks and another 10 more DataNodes will contain those replica too.

Date :

=> so, there will be 30 DataNodes in total to store blocks and ensure 3-levels of replication.

- we could have stored replicas in the same DataNode but then we could not have leveraged the parallel processing of distributed file system.

Ans.

answer – 9.1

1605023

Here, relations are —

Student(id, name, cgpa, yearAdmit)
Takes(id, course-id, semester, year)

and the query to be processed is —

```
Select r.id, course-id, cgpa
 from student r, takes s
where r.yearAdmit = s.year
```

Here, shared-nothing architecture is used. 10 nodes $n_1, n_2, ---, n_{10}$ are used to store r and s relations

by horizontal partition vector on

id = 121000, 131000, 141000, 151000, 161000, 171000, 181000, 191000, 201000.

Date :

So, there are 10 partitions in total into 10 given nodes.

r is horizontally partitioned into $r_1, r_2, ---, r_{10}$.

S is horizontally partitioned into $s_1, s_2, ---, s_{10}$.

Here, the range of yearAdmit is 2015 to 2021.

a.

Here are total 7 values for yearAdmit/year in the given r and s relations. So, one of the partitions will contain two year values depending on data distribution. The 6 partitions of student and takes using yearAdmit are as follows –

i. r_1' and s_1' ⟶ yearAdmit/year < 2016 (v_1)

ii. r_2' and s_2' ⟶ 2016(v_1) ≤ yearAdmit/year < 2017 (v_2)

iii. r_3' and s_3' ⟶ 2017(v_2) ≤ yearAdmit/year < 2018 (v_3)

iv. r_4' and s_4' ⟶ 2018 (v_3) ≤ yearAdmit/year < 2019 (v_4)

v. r_5' and s_5' ⟶ 2019 (v_4) ≤ yearAdmit/year < 2020 (v_5)

vi. r_6' and s_6' ⟶ 2020 (v_5) ≤ yearAdmit/year.

b.

performing the given query with aforementioned 6 nodes-

steps

i. Repartition $r_1, ---, r_{10}$ into $r_1', ---, r_6'$. Do the same for $s_1, ---, s_{10}$ to get $s_1', ---, s_6'$. (based on year.)

ii. assign r_i', s_i' partitions to node N_i.

iii. perform $r_i' \bowtie s_i'$ at node N_i.

Ans.

answer – 9.2

160523

We have the following relations –

i. Customer (id, name, type, country)

ii. Purchase (id, product-id, p-country, date)

We have to process the following query –

select r.id, product-id from customer r, purchase s where r.id=s.id

Here,

Shared-nothing architecture is used with nodes $N_1, ---, N_5$ where r and s are stored by horizontal partition vector on id = [1000, 2000, 3000, 4000].

So, we have 5 partitions in total into 5 nodes where –

i. r is horizontally partitioned into $r_1, ---, r_5$.

ii. s is horizontally partitioned into $s_1, ---, s_5$.

a. performing $r \bowtie_{r.id=s.id} s$ using provided 5 nodes –

steps

i. no repartitioning is required here.

ii. perform $r_i \bowtie s_i$ at Node N_i.

b.

Here, relations r and s are partitioned based on id and query attribute is also id. So, no repartitioning is required.

Ans.

answer – 10.1

1605023

• problem on External Sort-merge (sort merge).

Here, in the given example,
number of runs, $N = 4$ and number of memory blocks, $M = 3$.

a ✗

We need to find the size of the memory M' when the external sort (merge-sort) can be done in single pass.

In this case, as we have $N = 4$ runs (each containing 3 tuples) after run creation step, we will need a memory with size $M' = 5$ to accomplish the merge in one pass.
Then, 4 blocks of memory will be used as input buffer where each run will be assigned one memory block. The remaining block will serve as output buffer.
Again, if $M' = 5$, then we will have 3 runs each containing 5 tuples except for the last run. A run with blank slots does not create any problem. It will be considered as one run in the merging step.

$N' = 12/5 \approx 3.$

b. If the entire relation can be brought to memory simultaneously by increasing memory size, then we can apply any in-memory sorting algo. like quick sort on the relation. Usually, relations in DB are stored in disk for its size being huge. For sorting, we have to fetch a part of relation to memory from disk at a time. Hence, we can not apply in-memory sorting algo. directly on a relation.

So, memory size M' = size of relation.

c. The number of merge passes required depends largely on the memory size. If we can complete the sorting with fewer passes, then required time will be reduced. Thus, the sorting performance improves and becomes efficient.

Ans.

answer – 10.2

1605023

Given relation, Person (NID, name, DOB, street, city).
Here, the relation has 160 million tuples and is range-partitioned into 160 nodes using NID.
The following queries are to be processed:

a. find the list of persons sorted by NID in ascending order.
b. find the list of persons sorted by DOB in ascending order.

a. explanation of query processing

The tuples are range-partitioned based on sorting attribute (NID). So, tuples can be sorted locally in each node and concatenate the result to get output.

b. explanation of query processing

As the given relation has been partitioned based on attribute (NID) other than sorting attr. (DoB), we can sort it in two ways —

i. range-partitioning sort on DoB

ii. parallel external sort-merge algo.

Ans.

answer – 11.1

160523

Here, the 4 relations r_1, r_2, r_3, and r_4 are partitioned into 30 nodes $N_1, N_2, ---, N_{30}$.

We need to propose a query execution plan with intra-operation parallelism for individual joins and inter-operation pipeline parallelism.

a.

Here, the query is as follows –

$r_1 \bowtie r_2 \bowtie r_3 \bowtie r_4 = \{(r_1 \bowtie r_2) \bowtie r_3\} \bowtie r_4 = (temp1 \bowtie r_3) \bowtie r_4 = temp2 \bowtie r_4$.

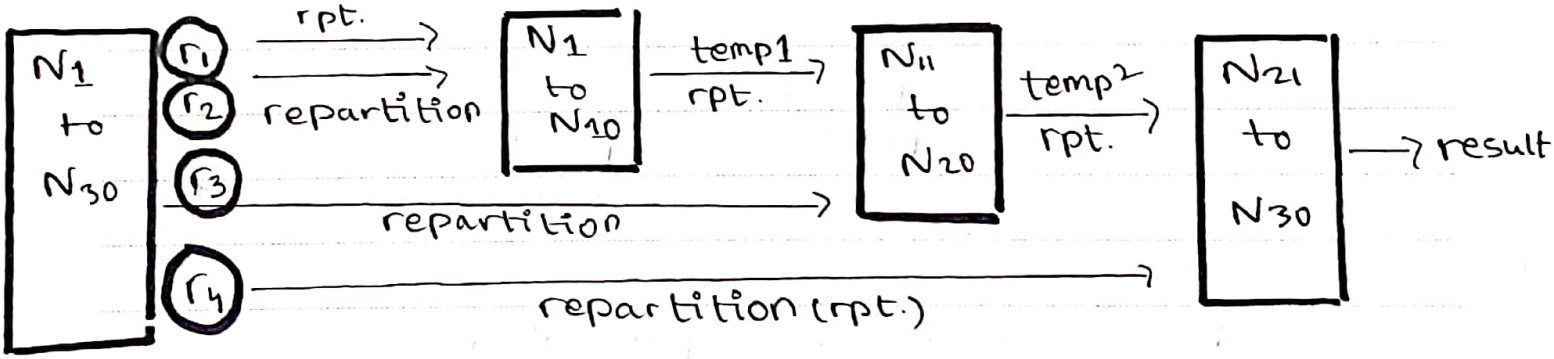
where,

- $r_1 \bowtie r_2$ ⟶ intra – operation parallelism involved (individual joins)
- $(temp1 \bowtie r_3)$ & $(temp2 \bowtie r_4)$ ⟶ inter-operation pipelined parallelism involved.

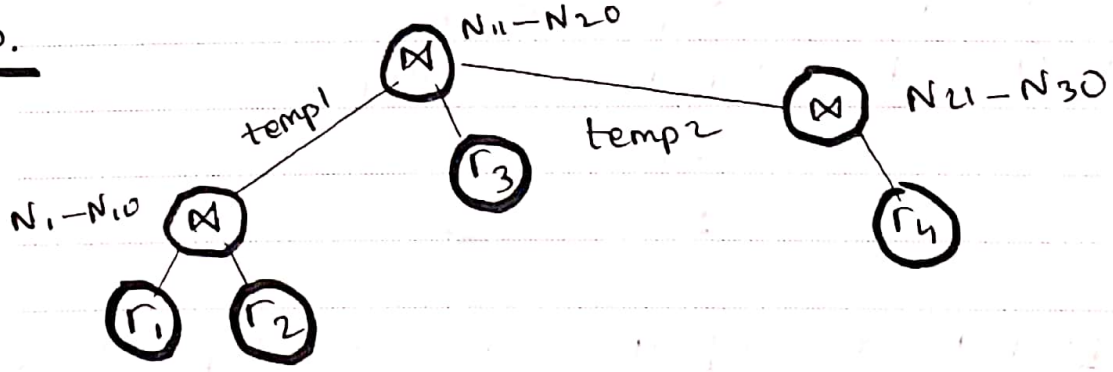
Here, among 30 nodes, $N_1 - N_{10}$ can be assigned to process the first joining ($r_1 \bowtie r_2$), $N_{11} - N_{20}$ can be assigned to process the second joining ($temp1 \bowtie r_3$), and the remaining 10 nodes can be assigned to process the third joining ($temp2 \bowtie r_4$).

So, we have to repartition r_1 and r_2 partitioned across 30 nodes to $N_1 - N_{10}$ based on query attribute. (intra-opt. parallelism)

We have to do the same repartitioning (based on query attribute) for pipelined output "temp1" and r_3 ($N_{11} - N_{20}$) as well as for pipelined output "temp2" and r_4 ($N_{21} - N_{30}$).



b.



Ans.

answer-11.2

1605623

a

we have to prepare query plan combining independent parallelism and pipelined parallelism for processing $S_1 \bowtie S_2 \bowtie S_3 \bowtie S_4 \bowtie S_5 \bowtie S_6$ where S_i $(1 \le i \le 6)$ = relations.

The nodes N_1, N_2, N_3, N_4, and N_5 are connected in a network and used to process the query.

Date :

Here, the joining will be independepent. for pairs of relations.
Hence, we have the following operations —

i. $temp1 = r_1 \bowtie r_2$ (N_1)
ii. $temp2 = r_3 \bowtie r_4$ (N_2)
iii. $temp3 = r_5 \bowtie r_6$ (N_3)

} independent parallelism

iv. $temp4 = temp1 \bowtie temp2$ (N_4)
v. $result = temp3 \bowtie temp4$ (N_5)

} pipelined parallelism.

b.

independent parallelism	pipeline parallelism
i. less useful in highly parallel system	i. same
ii. operation has to pipelinable	ii. same
iii. limited pipeline (can not use all nodes)	iii. same
iv. supports all types of joining	iv. supports only a limited portion of joining types
v. joining operation can not start before inputs from previous step are fully fetched. Hence, it works slower considering this aspect.	v. pipelining allows joining to start before inputs from previous step are fully fetched. Hence, it works faster considering this aspect.

Ans.

answer – 12.1

1605023

Given schema, person(NID, name, street, city, district, DOB, income).

And, nodes are $N_1, N_2, ---, N_{64}$.

Here, partitioning attribute is district and each node contains tuples of a single district.

Queries are —

i. Q_1 → select * from person where district = "Dhaka"

ii. Q_2 → select * from person where district = "Sherpur".

a.

Each of the queries Q_1 and Q_2 will be processed in single node because partitioning attribute and query attribute are same here.

b. There is a possible skewing which may introduce gap between processing time of two queries. It is highly likely that there are more tuples in "Dhaka" node than "Sherpur" node. As a result, query-1 takes more time than query-2 to complete. To improve processing time for Q_1, we can apply following skewness handling approaches —

i. repartition the relation based on other attribute

ii. partition tuples virtually (say, tuples in each node can be further virtually partitioned into 64 v.nodes and this will allow parallel query processing in 64 v.nodes for any district).

But, it will be tough to improve processing time if skewness is significantly high.

Ans.

answer - 12.2

1605023

given schema, person(NID, name, street, city, district, DOB, income).
And, nodes are $N_1, N_2, ---, N_{64}$.
Here, partitioning attribute is NID and there is no skew.
given,

Q_{21} ⟶ select *
from person
where NID > 4567891230 and NID < 8678912340.

Also,
N_{10} contains tuples with NID ranging between 4567891225 and 5000000000.
And, N_{30} contains tuples with NID ranging between 8678912340 and 8978912340.

Here, partitioning attribute and range query attribute are same. Hence, query will be processed at each node whose partition overlaps with the specified range of values ($N_{10} - N_{29}$ in this case). (parallel processing at each node)

Ans.

answer-12.3

1605023 Date:

Given schema, person (NID, name, street, city, district, DOB, income).

And, nodes are $N_1, N_2, ----, N_{64}$.

Here, partitioning attribute is district and each node contains tuples of a district.

The queries are —

i. Q_1 ⟶ select * from person

ii. Q_2 ⟶ select * from person where DOB > "01-Jan-2020".

reasons

Q_1 ⟶ no query attribute is provided.

Q_2 ⟶ partition attribute and range query attribute are not same.

Therefore, these queries will be processed in parallel in all nodes.

Ans.

1605023

answer – 12.4

given schema, person(NID, name, street, city, district, DOB, income).
And, nodes are $N_1, N_2, ---, N_{64}$.
Here, partitioning attribute is district and the given query is as follows –

Q31 ⟶ select income, count(NID)
from person
Group by income,

Steps

i. There will be skewing effect as tuples are partitioned based on "district". The relation will be repartitioned on grouping attr. "income".

ii. Then, the aggregation values "count" will be computed for each "income" value based on "NID".

possible optimization

consider the aggregation operation –

i. Perform aggr. operation at each node on those tuples stored in its local disk. This results in tuples with partial sums at each node.

ii. Result of the local aggregation is partitioned on the grouping attributes, and the aggregation performed again at each node to get the final result.

Ans.

answer – 12.4

1605023

given schema, Person (NID, name, street, city, district, DOB, income)

Here, person relation has 160 million tuples distributed among nodes $N_1, N_2, ---, N_{64}$. Partitioning is done on the attribute "district". So, each node contains tuples from one specific district. Also, repartition on income requires to transfer 80% tuples from each node. So, 20% tuples remain at same node after repartitioning.

The query is given as follows –

Q31 ⟶
```
select income, count (NID)
from person
group by income.
```

[a] This aggregate query will be processed in the following manner –

i. partition attribute ("district") and grouping attribute ("income") are different. So, person relation will be repartitioned based on "income" attribute.

ii. aggregate values "count (NID)" will be computed locally at each node.

[b] 80% of the total tuples are shifted and the rest of the tuples (20%) remain at same node after repartitioning.

∴ repartition cost = 160m × 0.8 = 128m tuples.

(c) optimization

We can reduce the cost of transfering tuples during partitioning by partial aggregation before partitioning.
It will be a distributive aggregate operation, that is, partial aggregation will take place at each node.
Thus, each node will contain locally aggregated tuples and these tuples will move among nodes (instead of main tuples of person relation) during repartitioning.
So, optimization is achieved.

Here, we have 640 distinct income values in person relation.
Assuming, each node has tuples with these 640 distinct income values. Then, if we perform aggregation at each node, each node will contain 640 aggregated tuples after operation.

Now, repartitioning these partially aggregated tuples based on grouping attr. "income" will result in each node having exactly (640 ÷ 64 =) 10 merged tuples with distinct income values.

optimized repartition cost

Here, 630 aggr. tuples among 640 of them leaves the corresponding node after repartitioning.
So, total (630 × 64 =) 40320 aggr. tuple moves among nodes during repartition.
∴ repartition cost = 630 × 64 = 40320 aggr. tuples
This is significantly less than previously calculated 118 Million tuples.
Thus, optimization is achieved.

Ans.

answer – 13.1 | 1605023 |

given schema,

person (NID, name, street, city, district, DOB, income).

Here,

person relation has 160 million tuples distributed among nodes $N_1, N_2, ---, N_{64}$. Partitioning attribute is "district".

query

Select * from person orderby income.

We need to explain how this SQL will be executed using exchange operator having the following parameters —

i. relation name
ii. partition type
iii. number of nodes (m)
iv. partition attribute

with the given sorting method.

[a] range partitioning sort

steps

i. person' = exchange_opt (person, "range-p", 64, "income").

Thus, person' (r_1', r_2', ---, r_{64}') relation will be repartitioned by exchange operator.

ii. Tuples at each node will be sorted locally afterwards.

[b] parallel external sort merge

steps

i. Tuples at each node will be sorted locally based on "income".

Date :

ii. person relation will be repartitioned and merged based on "income" by exchange operator afterwards.

Ans.

answer – 14.1

1605023

given schema,

person (NID, name, street, city, district, DOB, income)

Employee (eID, NID, organization, position, district)

Here,

Person relation has 160 million tuples and Employee relation has 10 million tuples. These tuples are distributed among 64 nodes ($N_1, N_2, ---, N_{64}$). Partitioning is done based on "district" attribute.

The query to be processed:

SQL ⟶
```
select *
from person p, employee e
where p.NID = e.NID
```

We need to explain how the query will be executed using exchange operator with partitioned join.

Explanation

The query will be processed through following steps:

i. $person' = (P_1', P_2', ---, P_{64}') = xchg(person, range, 64, NID)$.

ii. $employee' = (e_1', e_2', ---, e_{64}')$
$= xchg(employee, range, 64, NID)$.

iii. $P_i \bowtie e_i$ at N_i based on "NID"
for $i \in [1, 64]$.

Ans.

answer – 14.2

1605023

given schema,

Person (NID, name, street, city, district, DOB, income)
Employeel eID, NID, organization, position, district, salary).

Here,
person relation has 160 million tuples and employee relation has 10 million tuples. Tuples are distributed among nodes $N_1, N_2, ---, N_{64}$ where partitioning attribute is district.

The query to be processed:

SQL ⟶ select *
from person p, employee e
where p.income > e.salary

We need to explain how this query will be executed using exchange operator with AFR join.

Explanation

Here, person relation is significantly larger than employee relation. so, the query will be processed through asymmetric FR join. Both the relations are initially partitioned based on "district". The person relation will be repartitioned based on "income" by exchange operator at first. Then, exchange operator will broadcast the whole employee relation.
Thus, employee will be replicated at each node $N_1, N_2, --- N_{64}$ containing $P_1', P_2', ---, P_{64}'$ respectively.
Finally, non-equijoin (based on p.income > e.salary) will be applied between P_i' and employee at each node N_i.

Ans.

answer – 15.1

1605023

We need to explain the impact of skewing using the parallel cost model : $T = \max(T_1, T_2, \ldots, T_n)$ for range partition join of $r \bowtie s$.

Explanation

In "range" partitioning join, relations are at first repartitioned based on an attribute by exchange operator and then locally joined at each node.

Assuming, all nodes have similar processing capacity and exchange operator completes range partitioning task with no irregularity.

Then, execution time for each parallel partitioning task will be similar.

If this repartitioning introduces execution skew by making some nodes highly loaded with tuples, then local join at each node will take different time.

And, nodes with highly loaded data will take significant amount of time compared to other nodes.

In parallel cost model,

operational cost = max { execution time for each parallel process }.

As a result, the operational cost for the parallel join will be significantly high.

Ans.

answer – 16·1

1605023

Date :

Some objectives and applications of data analytics for a mobile phone company like TeleTalk, Grameen Phone, etc. are discussed:

objectives

i. customer satisfaction maximization

ii. profit maximization

applications

i. By analyzing customer profile features, billing pattern, calling pattern, and internet usage pattern, peak hours for different services can be learnt and predicted. User specific packages can be generated and offers can be introduced based on these information.

ii. By analyzing past history of sales, future sales can be predicted. The data can be further used to decide what/how much of a product to produce/stock. Also, customers can be targeted for increasing profit.

Ans.

answer – 16.2

1605023

(a) We have to explain the usage of federated database system or mediator system in the content of the management of higher education in Bangladesh by UGC.

Explanation

University Grants Commission (UGC) may generate and maintain different statistics concerning higher education across the country. Decisions will be made and policies will be imposed based on these stats.

Each institution has its own database of students, faculties, etc. which is maintained by various database management systems.

UGC can use federated DB or mediator system to integrate data from all these databases. Federated system supports global query and update whereas mediator system supports common query but not update. But, UGC can use either of the systems as they do not require to update information.

(b) We have to identify some entities and corresponding attributes for global schema for the above scenario.

Entities with Attributes

i. Student (CGPA, district, thana)

ii. Faculty (qualification), etc.

Ans.

answer—17.1

1605023

We have to design a star schema for the data warehouse of call duration and bill of different types of calls of different users from all mobile service phone companies.

The steps for designing a star schema are as follows:

Step – 1

Design fact table with measure attributes and dimensional attributes.

Step – 2

Design the dimensional tables.

Step – 3

Design the star schema.

Solution

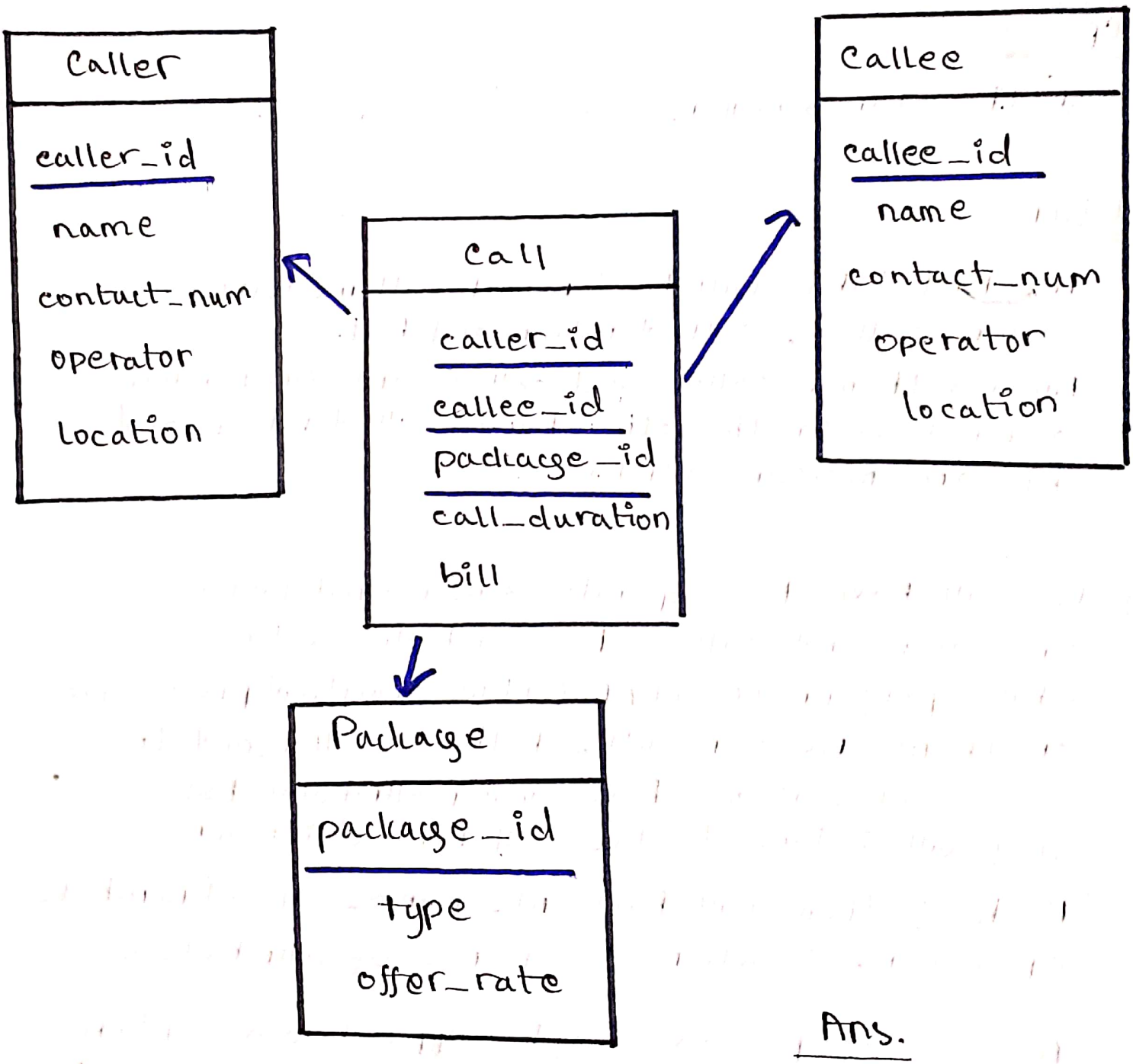
i. The fact table will be for each call involving caller, callee, call duration and bill. Among them, caller and callee are dimensional attributes. On the other hand, call duration and bill are measure attributes.

ii. We will have two separate dimensional tables for caller and callee to avoid difficulties while performing OLAP (online analytical processing) on them. As a result, both the caller and the callee schemas will have similar attributes but they will belong to two different dimensions. Both of them will have id, name, contact number, operator, location / tower info as attributes.

Also, some package may be applied to a certain call which regulates the pricing. So, an additional package attribute will be added to fact table

as dimensional attribute and a dimensional table for package will be added.

(iii.) Thus, we have the following star schema:



Ans.

answer – 18.1

1605023

We need to answer/provide solutions to the following questions:

a. Find four useful DSS reports that can be generated from the given star schema from aggregations (average, max, min, sum, count).

b. Find any missing dimension.

Solution (a)

DSS report is basically the output from a data warehouse structure following the star schema. This report is generated from aggregations (average, max, min, sum, count) rather than normal queries. DSS reports usually help policy makers in decision making as they provide overall statistics of stored data.

The followings are the four DSS reports that can be generated from given star schema:

i. Day or month or year-wise test count.

ii. gender or district-wise test count.

iii. average test prices

iv. pathologists' qualification-wise test count

These DSS reports can be generated from the provided dimension tables.

Thus, our policy makers can get clinical dss reports from national health data warehouse.

(b)

There is no dimension (table) for hospital data. As a result, report focusing on hospital's performance can not be generated.

Ans.

answer-18.2

1605023

(a) Write cross tabulation structure of sales by cloth_size and item_name.

Solution

color = all

cloth_size \ item_name	dress	pants	shirt	skirt	total
small	8	18	23	2	51
medium	12	6	8	5	31
large	15	3	18	0	36
total	35	27	49	7	118

(b) Write SQL to find the cross-tab.

Solution

SQL for color

```
select sum(quantity) from sales
```

SQL for total of item_name

```
select item_name, sum(quantity)
from sales
group by item_name
```

SQL for total of cloth_size

```
select cloth_size, sum(quantity)
from Sales
group by cloth_size
```

SQL for other cells

```
Select item_name, cloth_size, sum(quantity)
from Sales
group by item_name, cloth_size
```

Ans.

answer – 18.3

1605023

Write SQL for all DSS reports on location hierarchy.

Solution

report on city

R_c = select region, country, state, city, sum(quantity) as tot_c
from sales s, store t
where s.store_id = t.store_id
group by region, country, state, city

report on state

```
Rs = select region, country, state, sum(tot_c) as tot_s
     from Rc
     group by region, country, state
```

report on country

```
Rco = select region, country, sum(tot_s) as tot_co
      from Rs
      group by region, country
```

report on region

```
Rr = select region, sum(tot_co) as tot_r
     from Rco
     group by region
```

Ans.

answer – 19.1

1605023

Compare homogeneous and heterogeneous DDBMS in terms of storage, querying and transaction.

Solution

Storage

Data insertion and accession from global user to a certain local site is simple and feasible in homogeneous DDBMS. Data exchange among local sites is also feasible in homogeneous DDBMS. This is because every site has same DBMS. As a result, interfacing overhead is negligible. Thus, homogeneous DDBMS provides interoperability. On the other hand, different sites operate on different DBMS in heterogeneous DDBMS and this introduces a huge interfacing overhead. As a result, above-mentioned operations are not feasible for heterogeneous DDBMS.

querying

Due to interfacing overhead, querying is much slower and more complex in heterogeneous DDBMS than in homogeneous DDBMS.

Transaction

Interfacing overhead effects transaction in the similar way it effects querying.

Ans.

answer—19.2

1605023 Date :

Explain, the challenges in data transparency and transactional reliability in DDBMS.

Solution

challenges in transparency

There are 3 types of data transparency and each type has its own issues to be tackled.

(i.) network / distribution transparency

A global—local schema mapping is required so that no server location / address is needed for accessing database.

(ii) replication transparency

Schema mapping management is required.

(iii) fragmentation transparency

Data maintenance and user query processing are required.

challenges in reliability

The following issues are needed to be handled properly:

i. concurrency transparency by distributed concurrency control protocol
ii. failure atomicity by commit protocols
iii. data replication and parallelism
iv. Durability
v. lock management

Ans.

answer – 20.1

1605023

Site 3 has initiated transaction T to transfer Hc. 1000 from account P at site 4 to account Q at site 2. Write down the tasks of TC3.

Solution

tasks of TC3

i. TC3 starts the execution of the transaction T.

ii. There are two sub-transactions T_1 and T_2 here:

(T_1) update account P (withdraw Hc. 1000)

(T_2) update account Q (deposit Hc. 1000).

Here, TC3 assigns these two sub-transactions T_1 and T_2 to TM_4 and TM_2 respectively.

iii. TC3 coordinates the termination of all parallel sub-transactions (commitment or abortion of sub-transactions at sites 4 and 2).

Ans.

answer — 20.2

1605023

Date :

site 1 has initiated transaction T to transfer tk. 1000 from account A at site3 to account B at site2 and transaction T1 to add tk. 5000 to account C at site3. Write down the tasks of TM3.

Solution

tasks of TM3

i. There are, in total, two sub-transactions in which TM3 is involved:

(TA) update account A at site 3 (withdraw tk. 1000)

(Tc) update account C at site3 (deposit tk. 5000).

TM3 maintains logs for these sub-transactions for recovery purpose.

ii. TM3, along with TC1, coordinates the execution and termination of the transactions at site3.

Ans.

1605023

answer – 21.1

Why is commit protocol needed to commit the transaction T initiated by TC1 in given scenario? Explain.

Solution

If the entire system, considered here, was a centralized one, then a commit at site 1 would be enough to terminate the transaction. But, we are dealing with a distributed system. Hence, transection may fail or problems may arise at different sites at different points of time during execution. Therefore, commit protocols are necessary to ensure atomicity across sites.

Ans.

answer – 21.2

1605023

Write down the activities of the coordinator of site 1 in phase 1 and phase 2 for the case when all the sites including coordinator wants to commit.

Solution

activities of TC1

phase – 1

i. TC1 asks TM2 and TM3 to prepare to commit transaction T by adding record <prepare T> to log, forcing log for T to stable storage, and sending <prepare T> message to TM2 and TM3.

phase – 2

i. Upon receiving <ready T> messages from TM2 and TM3, TC1 adds a decision record <commit T> to log and forces log for T to stable storage.

ii. TC1, then, sends <commit T> message to TM2 and TM3 informing them of the decision.

Ans.

answer – 21.3

1605023

Write down the activities of the site3 in phase 1 and coordinator in phase 2 in the case when site3 does not want to commit.

Solution

activities of TM3 in phase 1

i. Upon receiving <prepare T> message from TC1, TM3 adds <no T> record to the log and sends <abort T> message back to TC1.

activities of TC1 in phase 2

i. Upon receiving <abort T> message from TM3, TC1 adds a decision record <abort T> to the log and forces log for T to stable storage.

ii. TC1 sends <abort T> message to TM2 and TM3 informing them of the decision afterwards.

Ans.

answer – 22.1

1605023

Site 3 has initiated transaction T3 to transfer Rs. 1000 from account P at site 4 to account Q at site 2. Explain the atomicity of T3 for different types of site and coordinator failures.

Solution

Here, TC3 is the coordinator and site 4 along with site 2 is the actively participating site. TM4 and TM2 manage the transaction T3 between accounts P and Q.

atomicity of T3 for site failure

In this case, site 4 or site 2 may fail during T3. when the failed site recovers, it examines its log to determine the fate of T3. If the log contains <commit T3> or <abort T3> record, then redo(T3) or undo (T3) operation must be carried out. If the log contains <ready T3> record, then the site must consult with TC3 to decide on the fate of T3. If none of the aforementioned scenario holds, then TC3 must abort T3 and site must carry out undo(T3) operation.

atomicity of T3 for coordinator failure

In this case, TC3 may fail during the execution of commit protocol for T3. Then, TM4 and TM2 must decide on T3's fate by looking into their respective

logs. If the log contains <commit T3> record, then manager must commit T3. If the log contains <abort T3> record, then manager must abort T3. If the log contains <ready T3> record, then manager must wait for TC3 to recover to find decision. If none of the above three records is in the log, then manager will abort T3.

Thus, atomicity is maintained for transaction T3 in different cases of failures.

Ans.

answer—23.1

1605023

There is an account A under organization 1 and another account B under organization 2.

a. Show all the steps to transfer tk. 5000 from account A to account B using persistent messaging protocol.

b. Discuss all types of failures and atomicity issues.

Solution

[a] Considering no failure during the entire process, the steps to complete transaction using persistent messaging protocol are as follows:

i. Transaction is initiated by account A and account A performs database update by withdrawing/subtracting tk. 5000 from the account.

ii. A transaction message is written and inserted in messages_to_send relation with a unique id.

iii. Message Delivery processor at site A generates the inserted message and sends it to site B.

iv. At site B, upon receiving message, message receiving processor executes transaction to add corresponding message to received_messages relation with a unique id. Assigning each message with unique id is a must to ensure that message for a specific transaction is written only once.

v. Account B processes the received message, updates database by depositing/adding tk. 5000 to its account, and marks the message as processed.

vi. Transaction is committed at site B.

vii. Acknowledgement message is sent to site A from site B.

viii. At site A, upon receiving the acknowledgement, the corresponding transaction message in messages_to_send relation is deleted.

ix. Site A commits the transaction.

Thus, atomicity is maintained throughout the whole process.

b failures and Atomicity Issues

i. If destination account B does not exist or is unreachable, failure message must be sent back to source account A.

ii. If sending transaction aborts, message must not be sent to site B.

iii. When transaction is aborted at site A or site A receives failure message, tk. 5000 must be deposited back in source account A. This process may require hooman intervention if account A is found to be closed.

Ans.

answer – 24.1

1605023

Put appropriate lock and show lock status.

Solution

T_1	T_2	T_3	T_4	Lock Status
LOCK-S(P)				GRANT(P, T_1)
READ(P)				
	LOCK-X(Q)			GRANT(Q, T_2)
	WRITE(Q)			
		LOCK-S(P)		GRANT(P, T_3)
		READ(P)		
			LOCK-X(P)	WAIT(P, T_4)
			WRITE(P)	

Ans.

answer – 24.2

1605023

a. given A = 100 and B = 100. Prove that the above concurrent schedule preserves database consisteney.

b. Explain conflict serializability of given schedule 2.

Solution

[a] If the value of A+B is preserved after the execution of transactions T_1 and T_2, then we can say that given concurrent schedule preserves DB consistency.

Here, at the beginning: A = 100 and B = 100.

i) After first block of operation on A by T_1: A = A − 50 = 50, B = 100.

ii) After first block of operation on A by T_2: A = A − 0.1·A = 45, B = 100.
Also, we have temp = A·0.1 = 5 in T_2 now.

iii) After second block of opt. on B by T_1: A = 45, B = B + 50 = 150.
T_1 commits at the end of this block of opt.

iv) After second block of opt. on B by T_2: A = 45, B = B + temp = 155.
T_2 commits at the end of this block of opt.

So, $A_{before} + B_{before} = 100 + 100 = 200$.
$A_{after} + B_{after} = 45 + 155 = 200$.

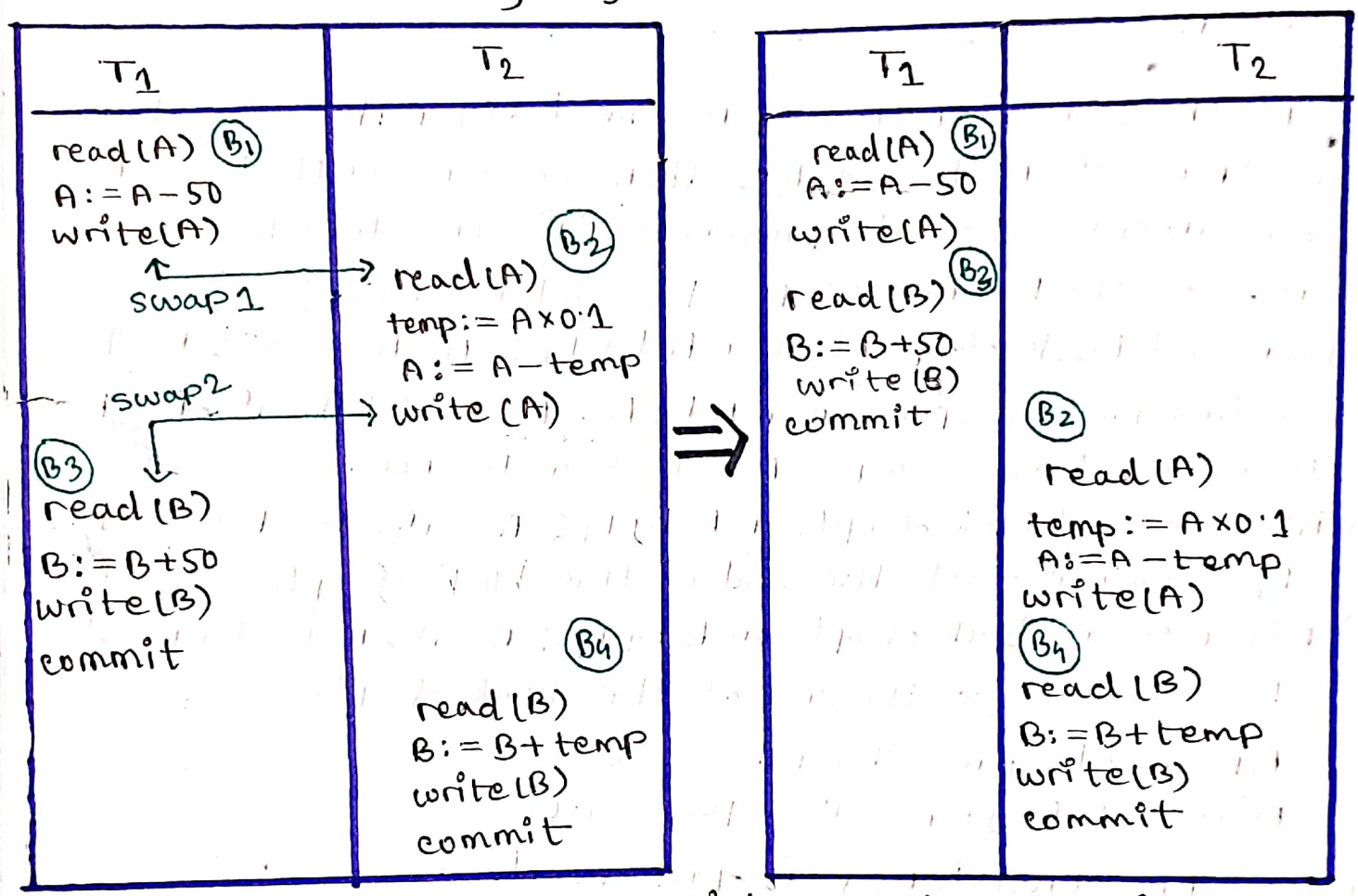
Thus, DB consistency is preserved.

[b] A confliet serializable schedule is a non-serial schedule which can be made a serial one by swapping non-conflicting instructions among participating transactions.

Here, schedule 1 is a serial schedule because T_2 starts after T_1 transaction's execution and commitment.

On the other hand, schedule 2 is not a serial schedule for its concurrent nature. But, schedule 2 is equivalent to serial schedule 1 as both of them yield same output after execution and preserve DB consistency.

Now, we can make the following changes in schedule 2 without introducing any conflict:



Here, swap 1 introduces conflict as read and write on same resource A.
swap 2 does not introduce conflict as read and write on two different resources A and B.

Thus, we can convert schedule 2 to schedule 1 by swapping block 2 and block 3 in execution timeline.

Hence, schedule 2 is conflict serializable as schedule 1 is a serial one. Ans.

answer – 15.1

160523

Let us consider the transaction T_{41} at site 4. The transaction displays of balance of account A. There is single lock manager at site 1.

i. Write the transaction with lock.

ii. Write the steps to obtain the lock by T_{41}.

Solution

(i) Transaction

i. LOCK-S (A)
ii. READ (A)
iii. DISPLAY (A)
iv. COMMIT (A)
v. UNLOCK (A)

A transaction does not remain serializable after commit operation. So, commit precedes unlock here.

(ii) Here, TC_4 initiates the transaction T_{41} at site 4. and sends lock request for data item A to single lock manager at site 1. Then, SLM forwards this lock request to all the sites having replica of data item A. So, site 1, site 2, and site 4 receive lock request for A from SLM. We need to remember that SLM can not make any decision about this transaction here. It just coordinates TC's and TM's of all the sites involved and manages locks. SLM does not have access to database or any particular data item. Rather, it uses

Date :

a dictionary (with mapping between available data items and sites where their corresponding replicas are available) to request lock for a particular data item to corresponding sites with replica. TM of the corresponding site basically decides on whether to give / grant lock for the data item. SLM, then, connects the Transaction coordinator at site4 (TC4) with the transaction manager (TM) to continue the transaction T41. Here, TM which responds and grants lock first involves in the later transaction process.

So, the steps are —

Steps

i. TC4 sends lock_S(A) request to SLM.
ii. SLM forwards lock_S(A) request to TM1, TM2, TM4.
iii. SLM forwards granted lock_S(A) from any one of those sites to TC4.

Ans.

answer - 25.2

1605023

Let us consider the transaction T_{21} at site2. The transaction adds tk.1000 to account B. There is single lock manager at site 1.

i. Write the transaction with lock.

ii. Write the steps to obtain the lock by T_{21}.

Solution

(i) Transaction

i. <START T_{21}>
ii. LOCK_X(B)
iii. READ(B)
iv. B := B + 1000
v. WRITE(B)
vi. COMMIT(B)
vii. UNLOCK(B)
viii. <END T_{21}>

WRITE(B) must be performed on both replicas of B at site3 and site4.

(ii) steps

i. TC2 sends lock_x(B) request to SLM at site1.
ii. SLM forwards lock_x(B) request to TM3 and TM4.
iii. Both TM3 and TM4 grant lock_x(B) request.

(In Read operation, lock grant from any one of the TM's is enough. But in Write operation, lock grant from all the TM's involved is a must for maintaining consistency in database.)

iv. SLM forwards granted lock_x(B) to TC2.

Ans.

answer – 26.1

1605023

A data item P is replicated in 12 sites. Transaction T_2 has Lock-x(P). Find the minimum number of messages required to

a. obtain this lock using majority protocol?

b. unlock using majority protocol?

Solution

(a) replica of data item P is available at 12 sites.

Hence,

messages for lock request = $\frac{12}{2} + 1 = 7$

messages for lock grant = $\frac{12}{2} + 1 = 7$

∴ minimum $(7+7) = 14$ messages are required to obtain LOCK-x(P) using majority protocol.

(b) Here,

messages for unlock = $\frac{\text{total replica}}{2} + 1 = \frac{12}{2} + 1 = 7$.

∴ minimum 7 messages are required to unlock using majority protocol.

Ans.

answer - 26.2

1605023

Data items A and B are replicated in 15 sites. Transaction T_{11} has LOCK-X(A) and LOCK-S(B).

a. What is the minimum number of sites required to obtain these locks using biased protocol?

b. Find the number of messages to obtain lock and release lock in each case.

c. Compare biased protocol with majority protocol in terms of performance.

Solution

(a) minimum number of sites required

- LOCK-X(A) ⟶ 15 sites.
- LOCK-S(B) ⟶ 1 sites.

(b) number of messages to obtain and release lock

- LOCK-X(A) ⟶ 2×15 or 30 msg to obtain lock.
  - ⟶ 15 msg to release lock.
- lock-S(B) ⟶ 2×1 or 2 msg to obtain lock.
  - ⟶ 1 msg to release lock.

(c)

majority protocol	biased protocol
more robust as transactions with both shared and exclusive locks can continue when majority of sites are up	less robust as transaction with exclusive lock fails when any one of the sites fails
same number of messages are required for both shared and exclusive locks which degrades overall performance	shared lock requires less number messages which eventually enhances overall performance
better performance in write/update-intensive transactions	better performance in read-intensive transactions

Ans.

answer – 26.3

1605023

The data item P is replicated in 8 sites $S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8$ and the weight of the sites are 2, 2, 2, 3, 3, 3, 4, 4 respectively.

a. Find Qw and Qr such that the protocol is
   i. biased ii. majority

b. Find the replicas to be locked for Qw and Qr other than (a) for the following cases:
   i. lock-s(P) ii. lock-x(P).

Here, the options are:

option – 1 → Starting the sites with lowest weights in ascending order.

option – 2 → Starting the sites with highest weights in descending order.

Solution

(a.) Here, $S = \sum_{i=1}^{8} w_i = 2+2+2+3+3+3+4+4 = 23.$

i. for biased protocol, lock-s(P) requires 1 replica lock. and lock-x(P) requires 8 replica lock.

So, Qr = 1 will be enough for lock-s(P)
and Qw = S = 23 will be enough for lock-x(P).
Here, $Qr + Qw = 24 > 23$ and $2 \times Qw = 46 > 23$.

ii. for majority protocol, lock-s(P) and lock-x(P) (both of them) require lock from at least $(\frac{8}{2}+1) = 5$ replicas.
Also, we can set Qw = Qr for the majority protocol case.
Considering 5 replicas with lowest weights will help us set Qr and Qw in such a way that quorum consensus protocol starts to act as majority protocol.

Therefore, $Qr = 2+2+2+3+3 = 12$ and $Qw = Qr = 12$ will be enough for lock-s(P) and lock-x(P) respectively.
Here, $Qr + Qw = 24 > 23$ and $2 \times Qw = 24 > 23$.

(b) Assuming, $Qr = 11$ and $Qw = 13$ where $Qr + Qw > S(23)$ and $2 \times Qw > S$.

i. Lock-S(P)

option – 1

weights = $w_1 + w_2 + w_3 + w_4 + w_5$
$= 2+2+2+3+3$
$= 12 > Qr(11)$.

So, sites to acquire locks from are S_1, S_2, S_3, S_4, S_5.

option – 2

weights = $w_8 + w_7 + w_6$
$= 4+4+3$
$= 11 = Qr$.

So, sites to acquire locks from are S_6, S_7, S_8.

ii. Lock-X(P)

option – 1

weights = $w_1 + w_2 + w_3 + w_4 + w_5 + w_6$
$= 2+2+2+3+3+3$
$= 15 > Qw(13)$.

So, sites to acquire locks from are $S_1, S_2, S_3, S_4, S_5, S_6$.

option – 2

weights = $w_8 + w_7 + w_6 + w_5$
$= 4+4+3+3$
$= 14 > Qw$.

So, sites to acquire locks from are S_5, S_6, S_7, S_8.

Ans.

answer – 27.1

1605023

construct a wait-for graph for the following.

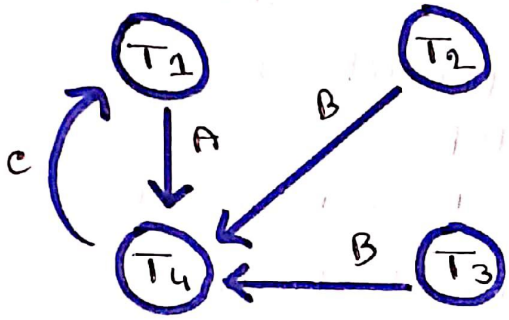
i. Transaction T_1 requests a lock for data A that is locked by T_4.

ii. Transaction T_2 requests a lock for data B that is locked by T_4.

iii. Transaction T_3 requests a lock for data B that is locked by T_4.

iv. Transaction T_4 requests a lock for data C that is locked by T_1.

Solution



Ans.

discussion – 27.1

why is site identifier for a transaction in LSB and local unique timestamp for that transaction in MSB of its global unique identifier? what would happen if it would be rerverse?

Solution

We want to put more bias on the timestamp value when we are considering the priority for resolving deadlock. Putting transactions with nearby timestamp value close to one another ensures the aforementioned bias. We can do this by putting local timestamp in MSB and site id in LSB of global id.

If we put site id in MSB and local timestamp in LSB, then more bias will be imposed on the site id when resolving deadlock and we simply do not want that.

Ans.

answer – 27.2

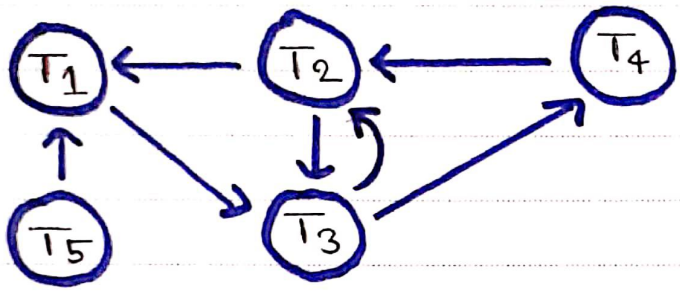
1605023

Date :

construct the global wait-for graph from the given local wait-for graphs for sites S_1, S_2 & S_3 and find the deadlock status.

Solution

global wait-for graph



Deadlock Status

There are, in total, 4 cycles in the global wait-for graph:

i. $T_1 \longrightarrow T_3 \longrightarrow T_2 \longrightarrow T_1$

ii. $T_1 \longrightarrow T_3 \longrightarrow T_4 \longrightarrow T_2 \longrightarrow T_1$

iii. $T_2 \longrightarrow T_3 \longrightarrow T_2$

iv. $T_2 \longrightarrow T_3 \longrightarrow T_4 \longrightarrow T_2$

These cycles indicate deadlock will arise during the execution of these transactions.

Ans.

answer – 28.1

1605023

a. Explain how the wide column representation supports flexible schema.

b. Compare RDBMS with wide column representation.

Solution

(a) In wide column representation, we can add new attributes to our tuples with no expensive operation. Even, a new attribute column can be added to one or two tuples and other tuples remain unchanged. These are possible with the usage of super columns and column families. Thus, wide column representation supports flexible schema.

(b)

Aspect	RDBMS	Wide Column
architecture	schema is fixed.	Schema is flexible.
storage management	simpler.	flexible schema makes it comparatively complex but still efficient for large volumn of data.
query processing	Simpler and faster for simple queries.	flexible schema makes it comparatively faster for complex queries.

Ans.

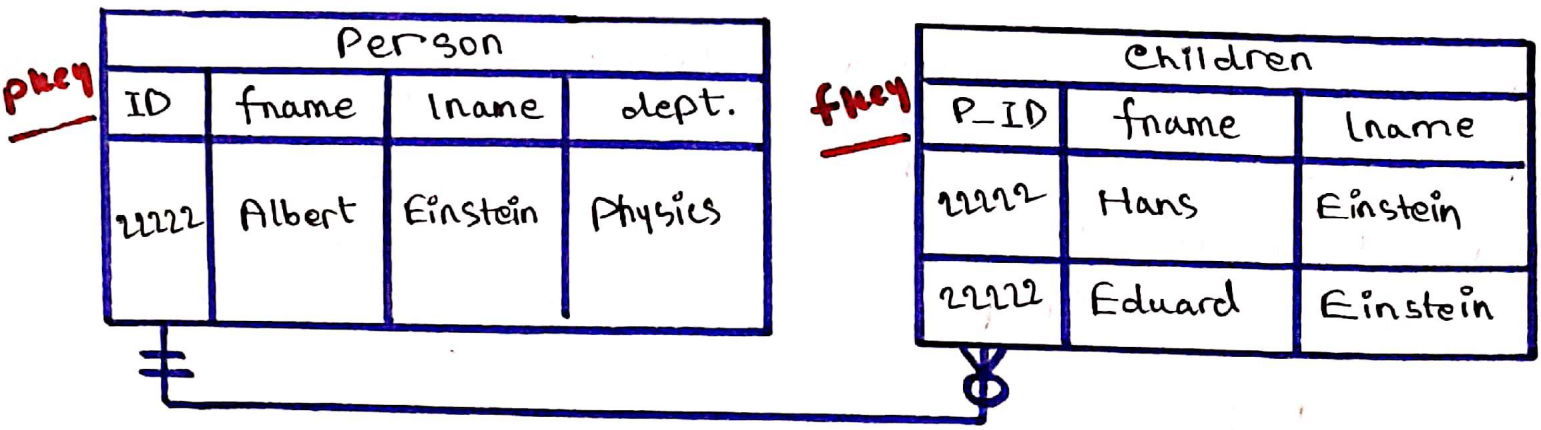
answer – 29.1

1605023

write the relational representation of the following JSON data:

```
{
    "ID": "22222",
    "name": {
        "firstname": "Albert",
        "lastname": "Einstein"
    },
    "department": "Physics",
    "children": [
        { "firstname": "Hans", "lastname": "Einstein"},
        { "firstname": "Eduard", "lastname": "Einstein"}
    ]
}
```

Solution



Ans.

answer – 30.1

1605023

a. Find the relational representation (schema) of this xml data.

b. Compare xml and relational models.

Solution (a)

Here, we have the following relations:

i. one-to-one relation between purchase order and purchaser

ii. one-to-one relation between purchase order and supplier

iii. many-to-many relation between purchase order and item.

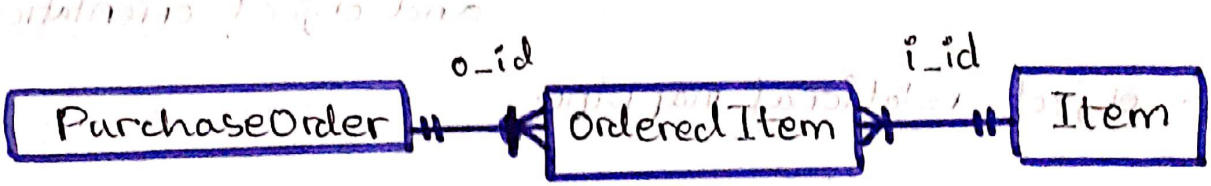
Considering these relations, we have found the following schemata:

i. PurchaseOrder (id, p_name, p_address, s_name, s_address, total cost)

ii. Item (id, description, per unit cost/price)

iii. orderedItem (o_id, i_id, quantity, total price)

We have the following relation here:



(b)

xml	relational models
i. schemaless	i. have schema
ii. no joining in query processing	ii. joining takes place
iii. occupies a lot of memory	iii. occupy relatively less memory

Ans.

answer – 30.2

1605023

a. Define SQL schema for the following using type inheritance.

A car-rental company maintains a database for all vehicles in its current fleet. For each vehicle, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color. Special data are included for certain types of vehicles:

i. Trucks: cargo, capacity.
ii. Sports Cars: horsepower, renter age requirement
iii. Vans: number of passengers
iv. Off-road vehicles: ground clearance, drivetrain (four- or two-wheel drive)

b. Insert one tuple in each table.

Solution

(a) At first, we will create a type "Vehicle" and then, several other types namely "Truck", "SportsCar", "Van", "off-road vehicle" under the type "vehicle".

=> create type Vehicle
(ID varchar(20) primary key,
L_num varchar(20),
manufacturer varchar(30),
model varchar(20),
P_date Date,
color varchar(20)) ref from(ID);

=> create type Truck under Vehicle
(cargo varchar(20),
capacity Number);

=> create type SportsCar under Vehicle
(horsepower Number,
r_age_req varchar(30));

=> create type Van under Vehicle
(p_capacity integer);

=> create type off_road vehicle under vehicle
(g_clearance Number,
drivetrain varchar(20));

Now, we will create a table "vehicles" of type "vehicle".

=> create table vehicles of vehicle ; following oracle syntax

Then, we will create four more tables corresponding to the different vehicle types using table inheritance of "vehicles".

=> create table trucks of Truck
under vehicles ;

=> create table sportscars of Sportscar
under vehicles ;

=> create table vans of Van
under vehicles ;

=> create table off-road vehicles of Off-road Vehicle
under vehicles ;

(b)

i. insert into Trucks(ID, l_num, ---, cargo, capacity)
values ("T1", "2301", ---, "medium", 2,000) ;

ii. insert into Sportscars(ID, l_num, ---, hp, r_age_req)
values ("T2", "2302", ---, 50, "whatever it is") ;

iii. insert into vans(ID, l_num, ---, p_capacity)
values ("T3", "2303", ----, 10) ;

iv. insert into Off-road vehicles(ID, l_num, ---, drivetrain)
values ("T4", "2304", ---, "whatever it is (2)") ;

Ans.

answer – 31.1

1605023

Select dept_name, head->name, head->address
from departments;

Write the output of the above SQL expression.

Solution

('CSE', 'Atique', 'CSE, BUET') or

```
{
  'dept_name': 'CSE',
  'head': {
     'name': 'Atique',
     'address': 'CSE, BUET'
  }
}
```

Ans.

answer – 31.2

Create type Customer with attributes c-id, name, street, and city. Use reference type for c-id. Insert the following records into customer table.

('c001', 'Arif', 'North', 'Dhalca')
('c002', 'Abdullah', 'South', 'Dhalca')

Create another type account with attributes acc-id, type, owner where owner is reference type and refers to customer table. Insert an account with acc-id = 'A001' and type = 'savings'. The owner of the account is Abdullah.

Implement the above using:

a. c-id as reference

b. system-generated reference

c. Compare the performance of the above implementation with the same with foreign key reference.

Solution

=> create type Customer
(c-id varchar(20) primary key,
name varchar(20),
street varchar(30),
city varchar(30))
ref from(c-id);

=> create table customers of Customer;

=> insert into customers
values ('c001','Arif','North','Dhaka');

=> insert into customers
values ('c002','Abdullah','South','Dhaka');

=> create type Account
(acc-id varchar(20) primary key,
type varchar(20), owner ref(Customer) scope customers);

=> create table accounts of Account;

(a) => insert into accounts values ('A001','savings','c002');

(b) => insert into accounts values ('A001','Savings',null);

=> update accounts
set owner = (select ref(c) from customers as c where c-id='c002')
where acc-id = 'A001';

(c) we have to bring two corresponding tables to memory and do joining operation in relational model where foreign key reference is used. On the other hand, we do not have to do any explicit joining in object-relational model since tuples are linked via references. Hence, we simply bring corresponding tuples to memory and carry out further operations. Therefore, the later is more efficient.

Ans.

recording ১৭৩৩

answer – 32.1

1605023

Compare DBMS query processing with IR query processing.

Solution

In DBMS SQL query, selection decision is made based on the true/false value of keyword comparison in "where" statement. Only those records/documents with true value are selected and fetched from DB.

On the other hand, in IR query, selection decision does not depend on the true/false value of keyword comparison. Instead, a relevance score is computed for each document against a keyword using certain formula. Then, a sorted order of documents based on the score is generated where doc with higher score gets priority in the list. This list is the IR search result.

Ans.

answer – 32.2

A document d is given as follows:

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be, government or private. For govt. employees, special attributes are ministry and designation. For non-govt. employees, special attributes are company name and position. National team is formed with the attributes as, id, team leader, game, organizing country, and date. The attribute team leader is reference type that refers to person. You have to define NID as reference.

a. Find TF(d, attribute)

b. Why logarithmic function has been considered instead of linear function in TF(d,t)?

Solution

(a) $TF(d, attribute) = \log\left(1 + \frac{n(d, attribute)}{n(d)}\right) = \log\left(1 + \frac{5}{86}\right)$.
(considering stop words as well)

(b) Consider a scenario where there are n occurrences of term t in doc d_1 and 2n occurrences of t in another doc d_2. We don't want to put 2x relevance on d_2 than d_1. So, we use logarithmic function to calculate $TF(d, t)$ instead of linear function.

Ans.

answer – 32.3

A document d_1 is given as follows:

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be government or private. For govt. employees, special attributes are ministry and designation. For non-govt. employees, special attributes are company name and position.

Another document d_2 is given as follows:

Define SQL schema for the following using type inheritance.
A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color.

a. Find IDF (attribute).

b. Find IDF (for).

Solution,

(a) $IDF(attribute) = \frac{1}{n(attribute)} = \frac{1}{1} = 1$

(b) $IDF(for) = \frac{1}{n(for)} = \frac{1}{2} = 0.5$

Ans.

answer – 32.4

1605023

A document d1 is given as follows:

A person has NID, name, street, city, thana, district and age. A person may be an employee with special attributes as salary and qualification (highest degree). An employee may be, government or private. For govt. employees, special attributes are ministry and designation. For non-govt. employees, special attributes are company name and position./

Another document d2 is given as follows:

Define SQL schema for the following using type inheritance.
A car-rental company maintains a database for all vehicles in its current fleet. For all vehicles, it includes the vehicle identification number, license number, manufacturer, model, date of purchase, and color./

a. Find relerance r(d1, {attribute, salary}).
b. Find relerance r(d2, {attribute, salary}).
c. Find ranking of the query {attribute, salary}.

Solution

(a) $r(d1, \{attribute, salary\}) = TF(d_1, attribute) \times IDF(attribute) + TF(d_1, salary) \times IDF(salary)$

$= \log\{1 + \frac{3}{52}\} + \log\{1+\frac{1}{52}\} = \log 55/52 + \log 53/52.$

(b) $r(d_2, \{attribute, salary\}) = TF(d_2, attribute) \times IDF(attribute) + TF(d_2, salary) \times IDF(salary)$

$= \log(1+\frac{0}{41}) \times 1/1 + \log(1+\frac{0}{41}) \times \frac{1}{1} = 0+0 = 0$

(c) Here, $r(d_1, \{attribute, salary\}) = \log\frac{55}{52} + \log\frac{53}{52}$.

$r(d_2, \{attribute, salary\}) = 0.$

So, $r_1 > r_2$ and $d_1 > d_2$. Ans.

answer – 33.1

1605023

Compare database query processing and information retrieval query processing.

Solution

In database query processing, SQL or SQL-like queries are written and only those records/tuples with true value in WHERE condition (for query keywords) are selected. On the other hand, in information retrieval query processing, all the records/tuples are ranked based on a relevance score w.r.t. query keywords. Therefore, fuzziness is observed in query result.

Ans.

answer – 33.2

A triangle has the coordinates (0,0), (5,0), (0,5). Represent the triangle by

a. lines b. points in the database.

Solution

a. {(0,0), (5,0), ID1}, {(5,0), (0,5), ID1}, {(0,5), (0,0), ID1}

b. {(0,0), ID1}, {(5,0), ID1}, {(0,5), ID1}

Ans.

answer – 33.3

1605023

Show the representation of the given polygon using a single tuple in relational model.

Solution

The tuple's representation in relational model for the given polygon :

$$\{(x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4), (x_5, y_5), ID_1\}$$

The database manager handles the variable length issue of these types of tuples.

Ans.

answer – 34.1

1605023

Explain how the polyhedrals (such as cube) can be represented by tetrahedrals.

Solution

We can represent any arbitrary polyhedral by dividing them into one or more tetrahedrals, just like triangulating polygons in two-dimensional space.

cube and tetrahedral are closely related. Piercing every other vertices of a cube so that no two are joined by an edge but any pair is joined by a diagonal of the cube's face begets a regular tetrahedral. All edges of that shape are equal and, therefore, all face angles are 60°.

Thus, a cube can be represented by in total of 5 tetrahedrals.

Ans.

1605023

answer – 35.1

If an application requires r reads and w writes per second, RAID 1 requires r+2w I/O operations per second. How?

Solution

In RAID 1, we can read a data block from any one of the disks. Also, we have to write to each disk for updating a data block. In RAID 1, a certain data block resides in a disk as well as in its mirror. Hence, in the afore-mentioned scenario, r+2w/sec I/O operations are required.

Ans.

answer – 35.2

If an application requires r reads and w writes per second, RAID 5 requires r+4w I/O operations per second. How?

Solution

In RAID 5, we read only the data block itself to extract some data. Also, we have to read the data block and its corresponding parity block into memory, update both of them, and then, write them back to disk for writing to a data block. Thus, in RAID 5, read operation involves 1 I/O operation and write operation involves 4 I/O operations. Therefore, in the aforementioned scenario, r+4w/sec I/O operations are required.

Ans.

answer – 36.1

1605023

Perform schema tuning for the above 2 applications.

Solution

Here, applications are –

a. Bank account verification system

b. National electronic health record system

(a.) Person – BAVS (NID, name, f-name, m-name, DOB, H-no, street, city, thana, district, division, income, profession, qualification, spouse – NID).

(b.) Person – NEHRS (NID, DOB, blood-group, height, weight, BMI, spouse – NID).

Ans.

answer – 36.2

a. Perform schema tuning for the above query by denormalization.

b. Explain the drawbacks.

c. How can these drawbacks be removed?

Solution

(a.) The normalized relations Student and Takes can be joined together to generate a denormalized relation Student-Takes with frequently accessed attributes from both normalized relations. The denormalized relation will, then, be stored separately and all the queries will be processed in this new relation.

b. More space is required in storage as the relations are denormalized. Also, extra operations are required to keep data consistency on updates across multiple relations.

c. Materialized views can be used to remove aforementioned drawbacks.

Ans.

answer—36.3

Explain how the join will be computed very efficiently in the above case.

Solution

Relations/tables that are joined frequently during query processing can be clustered together on the same disk block. Thus, number of seeks (disk head movement) gets reduced. Consequently, join operation takes less time and computation becomes very efficient.

Ans.

answer – 37.1

1605023

How can you improve the above queries?

Solution

The query is given as follows:

```
select sum(salary)
from instructor
where dept_name = ?
```

with

parameters {CSE}, {ME}, {CE}, --- from client side queries.

Here, each call from client to server for individual department has an overhead of network communication as well as query processing at server side. We can improve the performance of above queries by replacing them with the following one:

```
select dept_name, sum(salary)
from instructor
group by dept_name
```

Thus, client side gets the desired results from server side with just one call for all departments. Each department can fetch the required result from this query result at client side. So, number of calls from client to server is reduced to one. Consequently, overhead associated with network (bandwidth) load and database workload gets reduced.

Ans.

answer – 38.1

1605023

a. Lock space is exhausted in long update transactions. Explain.

b. Log space is exhausted in long update transactions. Explain.

c. Recovery time is increased in long update transactions. Explain.

Solution

(a.) In this long update transaction scenario, each data item (associated with one employee) requires individual x-lock, that is, lock-X(empN, TID). x-lock can not be released before the last commit in a specific transaction for a particular record in two-phase lock scheme. As a result, lock space gets exhausted in this scenario since lock manager grants lock for each data item.

(b.) In this long update transaction scenario, each data item has its own log record. As a result, huge amount of log records are stored during such a transaction. Also, they are not usually erased for recovery purpose. Thus, log space gets exhausted in this scenario.

(e.) In this long update transaction scenario, the whole thing is considered as one transaction with log records associated with individual employees. As a result, no checkpoint is set in-between any pair of log records. In fact, checkpoints are set at the beginning and at the end of this one gigantic transaction. In addition to that recovery manager takes recovery measures on the basis of these checkpoints. Thus, recovery time gets increased in this scenario.

Ans.

answer — 39.1

1605023

"The average throughput of system-A = (99+1)/2 = 50 tps and for system-B = (50+50)/2 = 50 tps" — justify it for the workload with an equal mixture of the two types T_1 and T_2 of transactions.

Solution

Here, for system-A, throughput of T_1 = 99 tps
throughput of T_2 = 1 tps

for system-B, throughput of T_1 = 50 tps
throughput of T_2 = 50 tps

For the given scenario, time-wise average throughput does not truly reflect the throughput of a system. Instead, task-wise average throughput serves as the actual reflection of the system's throughput. It can be computed using harmonic mean.

Hence,

average throughput of system-A = $\frac{2}{\frac{1}{99}+\frac{1}{1}}$ = 1.98 tps

average throughput of system-B = $\frac{2}{\frac{1}{50}+\frac{1}{50}}$ = 50 tps

Ans.

answer – 39.2

a. Why do TPC-H queries emphasize on aggregation?

b. Why does TPC-H prohibit materialized view?

Solution

(a) TPC-H benchmark suite is entirely dedicated for working with big data analytics in data warehouses. And, aggregation queries are the main operation in big data analytics. Hence, TPC-H queries emphasize on aggregation.

(b) Materialized view serves as the concise representation of fact table for particular query in data warehouse OLAP. Due to its reduced size, materialized view can be brought into memory and we can run in-memory operation on it. Further access to disk is not required during the operation. But, the target of a data warehouse is to work on big data with proper scalability. Hence, TPC-H prohibits usage of materialized views.

Ans.