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.