**Model 2023**

**I**

1. Let R and S be 2 relations. R has 10.000 records; a page can hold 10 R records.

S has 2.000 records; a page can hold 10 S records.

52 buffer pages are available.

Compute the cost of:

SELECT \*

FROM R INNER JOIN S

ON R.a = S.b

using page-oriented nested loops join and block nested loops join;

S is the outer relation.

⇒ we need to consider the number of I/O operations required for each join algorithm:

* Page-Oriented Nested Loops Join:

In this join algorithm, we iterate through each page of the outer relation (S) and for each page, we iterate through each record in the inner relation (R). The cost is determined by the number of I/O operations required to read and write the data.

Given:

* + R has 10,000 records with a page size of 10 records per page. So, R has 10000 / 10 = 1,000 pages.
  + S has 2,000 records with a page size of 10 records per page. So, S has 2000 / 10 = 200 pages.
  + 52 buffer pages are available.

The cost of page-oriented nested loops join can be calculated as follows:

* + Cost of reading the outer relation (S)

Since S is the outer relation, we need to read all its pages once.

So, the cost is equal to the number of pages of S.

Cost of reading S = 200 (pages)

* + Cost of reading the inner relation (R):

For each page of S, we need to read all pages of R.

Since there are 200 pages in S, we need to read 200 \* 1000 = 200000 pages of R.

Cost of reading R = 200000 (pages)

* + Total cost:

The total cost is the sum of the costs of reading the outer relation (S) and the inner relation (R).

Total cost of page-oriented nested loops join = Cost of reading S + Cost of reading R = 200 + 200000 = 200200 (pages) I/Os

* Block Nested Loops Join:

In this join algorithm, we read blocks of pages from the outer relation (S) and the inner relation (R) instead of individual pages. The cost is determined by the number of I/O operations required to read and write the data.

Given:

* + R has 10,000 records with a page size of 10 records per page. So, R has 10,000 / 10 = 1,000 pages.
  + S has 2,000 records with a page size of 10 records per page. So, S has 2,000 / 10 = 200 pages.
  + 52 buffer pages are available.

The cost of block nested loops join can be calculated as follows:

* + Cost of reading the outer relation (S):

Since S is the outer relation, we need to read all its pages once.

So, the cost is equal to the number of pages of S.

Cost of reading S = 200 (pages)

* + Cost of reading the inner relation (R):

For each block of S, we need to read all blocks of R.

The number of records that can be accommodated in a block is determined by the available buffer pages.

Number of records in a block = buffer pages \* page size = 52 \* 10 = 520 records

Number of blocks in R = Total records in R / records in a block = 10,000 / 520 ≈ 19.23 (blocks)

Since we cannot have a fraction of a block, we need to round it up to the nearest integer.

Number of blocks in R = 20 (blocks)

Cost of reading R = Number of blocks in R = 20 (blocks)

4 \* 1000 = 4000 I/Os

* + Total cost: The total cost is the sum of the costs of reading the outer relation (S) and the inner relation (R).

Total cost of block nested loops join = Cost of reading S + Cost of reading R = 200 + 20 = 220 (blocks)

block size: 50 ⇒ 200/50 = 4 S blocks

200 + 4 \* 1000 = 4200 I/Os

Therefore, the cost of the SELECT query using page-oriented nested loops join is 200200 I/Os, and the cost using block nested loops join is 4200 I/Os.

1. Let R and S be 2 relations.

R has 10.000 records; a page can hold 10 R records.

S has 2.000 records; a page can hold 10 S records.

Compute the cost of sorting R using external merge sort with 200 buffer pages.

1. Let R and S be 2 relations.

R has 10.000 records; a page can hold 10 R records.

S has 2.000 records; a page can hold 10 S records.

R is stored at București, S is stored at Cluj-Napoca.

Compute the cost of: SELECT \*

FROM R INNER JOIN S

ON R.a = S.b

using simple nested loops join (tuple-oriented) in Cluj-Napoca, without caching;

S is the outer relation.

* + td time to R / W a page from / to disk
  + ts time to ship a page
  + 200td + 2000 \* 1000 (td + ts) = 200td + 2000000 (td + ts)

1. Encode the data de gustibus non disputandum using the secret encryption key metallica and the table of codes below. Write the last 5 characters in the result.

**II**

1. T1 and T2 are 2 concurrent transactions, both active at time t.

Choose the correct answer(s): **a. The following execution describes a write read conflict: At time t, T2 is reading a data object previously written by T1.** b. The following execution describes a write read conflict: At time t, T2 is writing a data object previously read by T1. c. The following execution describes a read write conflict: At time t, T2 is reading a data object previously written by T1. d. The following execution describes a read write conflict: At time t, T2 is writing a data object previously read by T1. e. none of the above answers is correct.

In a write read conflict, one transaction writes a data object and another transaction reads the same data object. This creates a conflict because the reading transaction might see an inconsistent state if it reads the data object before the writing transaction commits its changes.

1. A schedule S:

a. is conflict serializable if and only if its precedence graph has exactly one cycle.

**b. is conflict serializable if and only if its precedence graph is acyclic.**

c. is conflict serializable if and only if its precedence graph has exactly two cycles.

d. is conflict serializable if and only if its precedence graph has exactly three cycles.

e. none of the above answers is correct.

Conflict serializability is a property of a schedule in concurrency control, indicating that the schedule is equivalent to some serial schedule. A schedule is conflict serializable if the order of conflicting operations (read-write or write-write) in the schedule can be rearranged to form a valid serial schedule without changing the outcome of the transactions.

The precedence graph represents the conflicts between operations in a schedule. It is a directed graph where the nodes represent the transactions, and the edges represent the conflicts between transactions. In the precedence graph, a cycle indicates a conflict that cannot be resolved, and therefore the schedule is not conflict serializable.

Options a, c, and d are incorrect because they specify a specific number of cycles in the precedence graph as a condition for conflict serializability, which is not accurate. The presence of any cycle in the precedence graph indicates a conflict that violates conflict serializability.

1. In SQL Server, under the READ UNCOMMITTED isolation level:

a. S locks must be acquired to perform read operations.

**b. read operations are performed without acquiring S locks.**

c. X locks must be acquired to perform write operations.

d. write operations are performed without acquiring X locks.

e. none of the above answers is correct.

Under the READ UNCOMMITTED isolation level in SQL Server (also known as the "dirty read" isolation level), read operations are not required to acquire shared (S) locks. This means that read operations can access uncommitted data or data modified by other transactions that have not yet been committed.

Option a is incorrect because S locks are not acquired for read operations under the READ UNCOMMITTED isolation level.

Option c is also incorrect because X locks (exclusive locks) are still required for write operations under the READ UNCOMMITTED isolation level. X locks ensure that write operations are exclusive and prevent other transactions from accessing or modifying the data until the changes are committed.

Option d is incorrect because write operations still require acquiring X locks under the READ UNCOMMITTED isolation level.

1. In horizontal fragmentation:

a. the reconstruction operator is the natural join.

b. the union of the horizontal fragments must be equal to the original relation.

c. fragmentation is performed with projection operators.

**d. fragmentation is performed with selection predicates.**

e. none of the above answers is correct.

In horizontal fragmentation, a relation is divided into multiple fragments based on a condition or selection predicate. Each fragment contains a subset of the original relation's rows that satisfy the specified condition.

Option a is incorrect because the reconstruction operator for horizontal fragmentation is the union, not the natural join. The natural join combines rows from different fragments based on matching attributes, but it is not used for reconstructing the original relation.

Option b is incorrect because the union of horizontal fragments does not have to be equal to the original relation. Fragments may contain subsets of the original relation's rows, and the union of these fragments may result in duplicate or missing rows compared to the original relation.

Option c is incorrect because horizontal fragmentation is not performed with projection operators. Projection operators are used to select specific attributes/columns from a relation, not to divide it into fragments.

Option d is correct because horizontal fragmentation is performed by applying selection predicates or conditions to divide the relation into fragments based on the specified criteria.

1. I is an index with search key <C1, C2, C3, C4>.

a. If I is a hash index, I matches condition C1 > 10 AND C2 > 7.

**b. If I is a hash index, I matches condition C1 = 10 AND C2 = 7 AND C3 = 1 AND C4 = 5.**

**c. If I is a B+ tree index, I matches condition C1 = 10 AND C2 = 7.**

**d. If I is a B+ tree index, I matches condition C2 = 7 AND C3 = 9.**

e. none of the above answers is correct.

A hash index is not suitable for range queries or conditions involving inequalities such as ">" or "<". Hash indexes are designed for equality searches, where the exact values of the indexed attributes need to match.

1. Let R be a relation with P pages. The cost of sorting R using simple two-way merge sort (i.e., with 3 pages in the buffer pool) is:

a. 𝜋^P

**b. 2P( log(4)P +1)**

c. 2P( log(2)P +1)

d. 2P( log(3)P +1)

e. none of the above answers is correct

To calculate the cost of sorting relation R using simple two-way merge sort, we need to consider the number of passes required and the number of I/O operations per pass.

In simple two-way merge sort, the relation is divided into sorted sublists that fit into the available buffer pages, and then these sublists are merged in subsequent passes until the entire relation is sorted.

Given:

* + R is a relation with P pages.
  + The buffer pool has 3 pages available.

The number of passes required can be calculated using the formula:

Number of passes = log(base B)((P + B - 1) / (B - 1))

In this case, B is the number of buffer pages, which is 3.

Number of passes = log(base 3)((P + 3 - 1) / (3 - 1)) = log(base 3)(P + 2)

Each pass involves reading and writing each page of the relation once. Therefore, the total number of I/O operations per pass is 2P.

The total cost of sorting is given by the formula:

Total cost = Number of passes \* I/O operations per pass = log(base 3)(P + 2) \* 2P = 2P(log(base 3)(P + 2))

To simplify the expression, we can convert the logarithm base from 3 to 2:

log(base 3)(P + 2) = log(base 2)(P + 2) / log(base 2)3

Therefore, the total cost becomes:

Total cost = 2P(log(base 3)(P + 2)) = 2P(log(base 2)(P + 2) / log(base 2)3) = 2P(log(2)(P + 2) / log(2)3) = 2P(log(2)(P + 2) / 1.585)

1. Consider the query: SELECT \* FROM R1, R2, R3 WHERE p1 AND p2 AND p3 The conditions tested by the predicates in the WHERE clause are statistically independent. The cardinality of a relation R is denoted by |R|. The reduction factor associated with predicate p is denoted by RF(p). The cardinality of the query’s result set can be estimated by:

a. |R1| ∗ |R2| ∗ |R3| / RF(p1) + RF(p2) + RF(p3)

**b. |R1| ∗ |R2| ∗ |R3| \* RF(p1) \* RF(p2) \* RF(p3)**

c. RF(p1) \* RF(p2) \* RF(p3) - ( |R1| + |R2| + |R3| )

d. |R1| + |R2| + |R3| + RF(p1) + RF(p2) + RF(p3)

e. none of the above answers is correct.

To estimate the cardinality of the query's result set, we need to consider the cardinalities of the involved relations (R1, R2, R3) and the reduction factors associated with the predicates (p1, p2, p3).

The reduction factor (RF) represents the selectivity of a predicate, indicating the fraction of tuples that satisfy the predicate.

In the given query, the conditions tested by the predicates in the WHERE clause are statistically independent. This means that the reduction factors can be multiplied together to estimate the selectivity of the entire WHERE clause.

The cardinality of the query's result set can be estimated by multiplying the cardinalities of the relations involved (R1, R2, R3) with the reduction factors of the predicates (p1, p2, p3). Therefore, the correct formula is:

|R1| \* |R2| \* |R3| \* RF(p1) \* RF(p2) \* RF(p3)

Let's evaluate the other options provided:

a. |R1| \* |R2| \* |R3| / RF(p1) + RF(p2) + RF(p3): This option incorrectly divides the product of the cardinalities by the sum of the reduction factors.

c. RF(p1) \* RF(p2) \* RF(p3) - (|R1| + |R2| + |R3|): This option subtracts the sum of the cardinalities from the product of the reduction factors, which does not estimate the cardinality correctly.

d. |R1| + |R2| + |R3| + RF(p1) + RF(p2) + RF(p3): This option adds the cardinalities and reduction factors together, which does not estimate the cardinality correctly.

1. Consider the execution below. When the system comes back up after the crash, it must ensure that:

a. T1, T3, T4 are durable; T2 and T5 are undone.

**b. T1, T3, T4 are undone; T2 and T5 are durable.**

c. T1 is undone only if T2 and T4 are also undone.

d. T2 is durable only if T5 is undone.

e. none of the above answers is correct.

In the given execution scenario, a crash occurs, which means that the system abruptly halts and loses any changes that were not durably stored.

Based on the execution, we can determine the durability of each transaction:

* + T1: T1 has completed before the crash, so it is considered durable.
  + T2: T2 was in progress at the time of the crash, so it needs to be undone because its changes may not have been durably stored.
  + T3: T3 was in progress at the time of the crash, so it needs to be undone because its changes may not have been durably stored.
  + T4: T4 was in progress at the time of the crash, so it needs to be undone because its changes may not have been durably stored.
  + T5: T5 has completed before the crash, so it is considered durable.

1. In data replication:

**a. primary site replication is an asynchronous replication technique.**

b. primary site replication is a synchronous replication technique.

**c. read-any write-all is a synchronous replication technique.**

d. read-any write-all is an asynchronous replication technique.

e. none of the above answers is correct.

1. A database access request contains:

**a. the requesting user.**

b. the criminal record of the requesting user.

**c. the operation the user wants to perform.**

**d. the requested object.**

e. none of the above answers is correct

A database access request typically includes information such as the requesting user, the operation the user wants to perform, and the requested object. However, it does not include the criminal record of the requesting user.

Therefore, none of the options provided in a, b, c, or d accurately represent the complete contents of a database access request. The correct answer is e. none of the above answers is correct.

1. Consider schedule S below over transactions T1, T2, T3, T4 (all transactions commit):

a. S is conflict serializable.

**b. S is not conflict serializable.**

c. (R(T4, C), R(T1, C)) belongs to the conflict relation of S.

**d. (W(T1, A), R(T2, A)) belongs to the conflict relation of S.**

e. none of the above answers is correct.

To determine if schedule S is conflict serializable, we can construct the precedence graph for the schedule and check for cycles. If the precedence graph has no cycles, then the schedule is conflict serializable.

The precedence graph for schedule S is as follows:

T1 -> T2 -> T3 -> T4

Based on the precedence graph, we can see that there is a cycle present (T4 -> T1). Therefore, schedule S is not conflict serializable.

Now, let's consider the conflict relation, which contains pairs of conflicting operations. A pair of operations is conflicting if they access the same data item, and at least one of them is a write operation.

Looking at the schedule:

(W(T1, A), R(T2, A)) is a pair of conflicting operations because they both access data item A, and W(T1, A) is a write operation.

Therefore, the pair (W(T1, A), R(T2, A)) belongs to the conflict relation of schedule S.

The options a, c, and e are incorrect because schedule S is not conflict serializable, and (R(T4, C), R(T1, C)) is not a conflicting pair in the schedule.

Therefore, the correct answers are b. S is not conflict serializable and d. (W(T1, A), R(T2, A)) belongs to the conflict relation of S.