IN3020&IN4020 – Database Systems Exam, 2025

4 June, 9:00am–1:00pm (Inspera)

Model Solutions

Thresholds

A:	80–100%
B:	65–79%
C:	50–64%
D:	35–49%
E:	20–34%
F:	0–19%

Marking guidelines

For each part of each question:

- minor mistake: 6% for 8% questions, 7% for 9% questions
- major mistake: 3% for 8% questions, 4% for 9% questions
- understood the question: 2% for all questions

Intermediate results (-1%, -3%) are also possible

Note: several exam questions allow for many correct answers, in which cases the solutions below give only one possibility.

Question 1 of 12 (9%)

Consider the following relations:

Table1:

A	B	C
0	1	1
0	1	2
0	1	*NULL*
1	*NULL*	1
1	1	2
1	1	*NULL*
2	*NULL*	1
2	1	2
2	1	3

Table2:

A
0
1
2

Explain which of the following (bag) RA expressions over these relations are defined (e.g., produce a relation), and why:

$$(\mathsf{Table1} \div \mathsf{Table2}) \times \mathsf{Table2} \tag{1}$$

and

$$(\mathsf{Table1} \times \mathsf{Table2}) \div \mathsf{Table2} \tag{2}$$

Evaluate the expressions that are defined (you do not need to show any intermediate steps; only the final results are required).

Solution

The first expression is defined: the division is defined since each attribute of Table2 is an attribute of Table1, and the Cartesian product is defined since the attributes of the result of the division, B and C, are disjoint with the attributes of Table2.

The second expression is not defined, because the arguments of the Cartesian product do not have disjoint attributes.

The result of the first expression is

Expr1:

A	B	C
0	1	2
1	1	2
2	1	2

Question 2 of 12 (8%)

In the lectures, we defined the left semi-join only for sets. Let us extend it to bag relations as follows: $R \ltimes S$ contains each tuple t from R that joins with at least one tuple in S; moreover, the multiplicity of t in $R \ltimes S$ is equal to its multiplicity in R. Express this left semi-join using the bag RA operations defined in the lectures.

Note: in this and the following questions you may introduce and then use your own notation for RA operations; for example, you may write 'I use LSJ for left semi-join below' and then use LSJ in place of $\ltimes$.

Solution

One possibility:

$$R \bowtie \delta(\pi^{\mathsf{b}}_{AttrRS}(S))$$

where $AttrR$ are the attributes of R and $AttrRS$ are the common attributes of R and S.

Question 3 of 12 (9%)

Consider relations with schemas Table1 [A, B], Table2 [C, D], and Table3 [E, F]. Compile the following SQL query to a (logical) query plan:

```
SELECT DISTINCT A FROM Table1 WHERE
        NOT EXISTS (SELECT * FROM Table2 WHERE
                EXISTS (SELECT * FROM Table3 WHERE C = E))
```

You may use the bag extension of the left semi-join defined in Question 2, as well as analogously defined extensions of the right semi-join and the anti-joins.

Note: you only need to write some query plan that does the same as the query; in particular, you do not need to show any intermediate steps of compilation or follow any specific compilation algorithm.

Solution

One possibility:

$$\delta(\pi_A(\mathsf{Table1} \triangleright (\mathsf{Table2} \bowtie_{C=E} \mathsf{Table3})))$$

Question 4 of 12 (8%)

Consider a relation with schema Table1 [A, B, C, D, E], where A is a primary key and each attribute occupies 10B. Let this relation has 1000 tuples and is stored in a continuous unspanned manner in blocks of size 10000B. Suppose we use a three-level primary index on attribute A that is sparse on all three levels, where a pointer also occupies 10B (and all index levels use blocks of the same size as the relation). How many block accesses are required, in the worst case, to retrieve all the tuples with a given search key using this index?

Solution

4

Question 5 of 12 (8%)

Recall that we defined the cost of a query plan Expr as the sum of estimated sizes of the results of its intermediate sub-plans. Suppose sub-plans Expr' and Expr'' of Expr are such that Expr'' is itself a sub-plan of Expr'. Assume that both sub-plans have $\bowtie$ as their top-level operation, so that their size estimations rely on the value containment and value preservation assumptions. In general, which scenario yields better cost estimation: when these assumptions hold in reality only for Expr', or only for Expr''? (You may also argue that both are equally important.) Justify your answer in one or two sentences.

Solution

Only for Expr'' is better, because the estimation of the number of tuples in the result of Expr'' contributes to the final cost of the plan in two ways: as a factor of the summand corresponding to Expr'', and as a base for estimating all the summands for the super-expressions of Expr'', including Expr'; on the other hand, the estimation of the number of tuples of Expr' does not affect the estimation of Expr''.

Question 6 of 12 (8%)

Recall that our two-pass hash-based natural join algorithm requires that

$$\min(\mathit{Blocks}(R_i), \mathit{Blocks}(S_i)) \leq \mathit{MMBuffers} - 1$$

for every bucket i, where R_i and S_i are the parts of the input relations R and S in bucket i. Propose a reasonable improvement of the algorithm that would allow it to handle cases where $\min(\mathit{Blocks}(R_i), \mathit{Blocks}(S_i))$ is slightly larger than $\mathit{MMBuffers} - 1$ for some buckets i.

Solution

We can apply nested-loop natural join on each such bucket (if the smaller size is just slightly larger than $\mathit{MMBuffers} - 1$, then the cost increase is also not large).

Question 7 of 12 (8%)

Consider the following three schedules:

$$S_1 : \mathtt{r}_2(X), \mathtt{w}_1(X), \mathtt{w}_2(X), \mathtt{r}_2(Y), \mathtt{r}_3(Y), \mathtt{w}_2(X), \mathtt{c}_2, \mathtt{c}_1, \mathtt{c}_3$$
$$S_2 : \mathtt{r}_2(X), \mathtt{w}_1(X), \mathtt{c}_1, \mathtt{w}_2(X), \mathtt{r}_2(Y), \mathtt{w}_2(X), \mathtt{c}_2, \mathtt{r}_3(Y), \mathtt{c}_3$$
$$S_3 : \mathtt{r}_2(X), \mathtt{w}_2(X), \mathtt{w}_1(X), \mathtt{c}_1, \mathtt{r}_3(Y), \mathtt{r}_2(Y), \mathtt{c}_3, \mathtt{w}_2(X), \mathtt{c}_2$$

Which pairs of these schedules are view equivalent? Explain your answer (try to be as concise and concrete as possible).

Solution

All three are view equivalent: all reads are from the same place (original values of X and Y), and the last write of X is the same (the second write of T_2).

IN3020&IN4020 – Database Systems Exam, 2025

4 June, 9:00am–1:00pm (Inspera)

Question 8 of 12 (8%)

Consider the following two transactions:

$$T_1 : \mathtt{r}(X), \mathtt{r}(Y), \mathtt{r}(Z), \mathtt{w}(Z)$$
$$T_2 : \mathtt{r}(X), \mathtt{w}(U), \mathtt{w}(Z), \mathtt{r}(U)$$

Assume that we use the two-phase locking protocol with wound-wait deadlock prevention. Write down a partial schedule of (the prefixes of) these transactions where T_1 has to abort due to the deadlock prevention policy. Your schedule should include appropriate locks and other relevant operations of the transactions.

Solution

One possibility:

$$\mathtt{b}_2, \mathtt{b}_1, \mathtt{l}_1(X), \mathtt{r}_1(X), \mathtt{l}_2(X), \mathtt{a}_1$$

Question 9 of 12 (9%)

Give an example of a partial schedule of two transactions such that

- each transaction both writes and reads the same data item X;
- the schedule has to abort one of the transactions at the end under the strict timestamp-ordering protocol without Thomas write rule, but can proceed under the same protocol with this rule.

Solution

One possibility:

$$\mathtt{b}_1, \mathtt{r}_1(X), \mathtt{b}_2, \mathtt{r}_2(X), \mathtt{w}_2(X), \mathtt{w}_1(X)$$

Question 10 of 12 (8%)

Consider the following schedule:

$$S : \mathtt{r}_1(X), \mathtt{w}_1(X), \mathtt{r}_2(Y), \mathtt{w}_1(Y), \mathtt{a}_1, \mathtt{r}_2(X), \mathtt{a}_2$$

Is this schedule recoverable? Is it cascadeless? If it is not cascadeless, modify it in a minimal way to make it cascadeless.

Solution

The schedule is cascadeless and hence recoverable.

Question 11 of 12 (8%)

In no more than three sentences, explain the main difference between the deferred update and immediate update recovery protocols, and why this difference is advantageous.

Solution

Deferred update protocol requires all dirty buffers changed by a transaction to be pinned (i.e., not allowed to flush) until the commit of the transaction, while immediate update protocol does not have this requirement. On the other hand, deferred update needs only REDO log entries, whole immediate update needs both REDO and UNDO. So, the (small) advantage of deferred update is that the log is smaller in size, but the (much larger) advantage of the immediate update is that it has no restrictions on the cache use. (Due to a slight ambiguity in the question formulation, it is enough to mention only the first difference and one of the two advantages for full mark.)

Question 12 of 12 (9%)

For brevity, assume that RDF graphs use only URIs (i.e., mention no blank nodes or literals). Let $f(D)$ be the direct translation of a database D with a single relation over the schema $\mathsf{Triple}[Subject, Predicate, Object]$ (where all attributes have type URI) into an RDF graph. Let $g(R)$ be a direct translation of a (bag) relation $\mathsf{R}[Att_1, \ldots, Att_N]$ (with all attributes of type URI) into a bag of mappings with domain $?Att_1, \ldots, ?Att_N$.

Consider the following SQL query Q:

```
SELECT t1.Subject
 FROM Triple t1 JOIN Triple t2 ON t1.Object = t2.Subject
 WHERE t1.Predicate = <http://example.org/marriedTo>
                                        AND t1.Subject <> t2.Object
```

Write a SPARQL query Q' that is equivalent to Q—that is, such that for every database D with a single relation Triple, we have that $g(Q(D)) = Q'(f(D))$.

Solution

For example,

```
SELECT ?x
    WHERE { ?x <http://example.org/marriedTo> ?y . ?y ?z ?u .
                                                FILTER ?x <> ?u }
```