IN3020&IN4020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Model Solutions

Thresholds

A:	76–100%
B:	63–75%
C:	50–62%
D:	37–49%
E:	24–36%
F:	0–23%

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.

bigskip

Question 1 of 12 (9%)

Let us call a unary set relational algebra operation $Op1$ *monotonic* if, whenever every tuple in relation $R1$ is also a tuple in relation $R2$ (over the same schema), every tuple in $Op1(R1)$ is also a tuple in $Op1(R2)$. Analogously, a binary set RA operation $Op2$ is *monotonic* if, whenever every tuple in relation $R1$ is also a tuple in relation $R2$ and every tuple in relation $S1$ is also a tuple in relation $S2$, every tuple in $Op2(R1, S1)$ is also a tuple in $Op2(R2, S2)$.

Which of the following operations are not monotonic: selection, projection, union, difference, Cartesian product, renaming, natural join, theta-join, left semi-join, left anti-join, division, generalised projection, left outer join? List only those operations that are not monotonic. For each of them, explain briefly (i.e., in one sentence) why you think so (you do not need to be formal, intuition is enough).

Solution

The following operations in the list are not monotonic:

- difference, because more tuples in the second argument may result in less tuples in the result,
- left anti-join, by the same reasons,
- division, by the same reasons,
- left outer join, because an additional tuple in the second argument may result in replacing tuples with nulls with tuples with non-nulls in the result.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 2 of 12 (8%)

Consider a relation over schema $R[A, B]$. Assuming that R may be a bag (multiset), write a bag relational algebra expression that outputs the relation over the same schema that contains all the tuples that have multiplicity exactly one in the input R (and nothing more).

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

Many solutions are possible; two possibilities are

$$\pi^b_{A,B}(\sigma_{D=1}(\gamma_{A,B,COUNT(C)\to D}(\pi^b_{A,B,1\to C}(R))))$$

and

$$\delta(R)\backslash(R\backslash\delta(R)).$$

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 3 of 12 (9%)

Consider the relation schema $Links[Source, Target, Length]$ and the following SQL query (that uses recursion):

```
WITH RECURSIVE Rec(Source, Target, Length, Hops) AS (
    (SELECT Source, Target, Length, 0 FROM Links)
          UNION
    (SELECT R.Source, L.Target, R.Length + L.Length, R.Hops + 1
     FROM Rec R, Links L
     WHERE R.Target = L.Source AND R.Hops < 2)
) SELECT DISTINCT Source, Target, Length FROM Rec;
```

Can this query be compiled to a logical query plan in the (bag) relational algebra considered in the lectures (i.e., the one without recursion)? If your answer is no, explain why. If your answer is yes, give such a plan.

Solution

Yes, it can. One possibility:

$$
\begin{array}{l}
\pi_{Source,Target,Length,0}(R) \\
\quad \cup \\
(\pi_{S\to Source,T'\to Target,L+L'\to Length,1}(\\
\qquad \pi_{Source\to S,Target\to T,Length\to L}(R) \bowtie_{T=S'} \pi_{Source\to S',Target\to T',Length\to L'}(R)) \\
\quad \cup \\
(\pi_{S\to Source,T'\to Target,L+L'+L''\to Length,2}(\\
\qquad \pi_{Source\to S,Target\to T,Length\to L}(R) \bowtie_{T=S'} \pi_{Source\to S',Target\to T',Length\to L'}(R) \\
\qquad\qquad \bowtie_{T'=S''} \pi_{Source\to S'',Target\to T'',Length\to L''}(R)))
\end{array}
$$

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 4 of 12 (8%)

Does there exist a relation and an attribute in this relation such that it is better not to use any index of any type considered in the lectures for exact search rather than to use one? Justify your answer in no more than a few sentences.

Solution

Yes, for example a relation with one attribute and one tuple in this relation. In this case, using an index requires two block reads and not using it only one.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 5 of 12 (8%)

Let the query compilation result in the query plan $\sigma_{A=a}(\sigma_{B=1}\mathsf{Table})$, and let the data consist of the following relation, where each attribute takes 2 bytes:

Table:

A	B
a	1
b	1
b	2
b	3

Optimise this query plan using the procedure considered in the lectures. In particular, give all query plans considered during the optimisation, compute their costs (no details of the computation are necessary), and determine the best plan according to the procedure.

Solution

A possible solution is that there are two query plans, the first is the initial $\sigma_{A=a}(\sigma_{B=1}\mathsf{Table})$ and the second is $\sigma_{B=1}(\sigma_{A=a}\mathsf{Table})$. The cost of the first is $8/3$ and the cost of the second is $8/2$. Thus, the best is the first (initial).

However, one may also claim that $\sigma_{A=a\&B=1}\mathsf{Table})$ is also a query plan, which has cost 0 and so is the best. This should also be accepted as a correct solution, provided the cost of the initial plan is computed as above.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 6 of 12 (8%)

Let us have 500 buffers in main memory and two relations, Table1 and Table2, with one common attribute, A. Let Table1 take 1000 blocks and Table2 take 500 blocks on disk. Finally, let $Vals(\mathsf{Table1.A}) = Vals(\mathsf{Table2.A}) = 1$. Which of the natural join algorithms considered in the lectures should be used to compute the natural join of Table1 and Table2, and why? Here, we assume that we should use the algorithm that is the most efficient according to our cost—that is, the one that minimises the number of disk reads and writes.

Solution

The only algorithm in the lectures applicable here is nested loop natural join, because the number of blocks for the only join value is too large for one pass for this join value and so no other considered algorithms are applicable due to their constraints.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 7 of 12 (8%)

Which of the following is true?

1. Two different serial schedules of two transactions are always conflict-equivalent.
2. Two different serial schedules of two transactions are never conflict-equivalent.
3. Two different serial schedules of two transactions may or may not be conflict-equivalent.

Justify your answer in at most 3 sentences.

Solution

The third is true. The two schedules are conflict-equivalent only if there are no conflicts at all (i.e., there is no data item such that at least one of the transactions writes it) and not conflict-equivalent otherwise.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 8 of 12 (8%)

Consider the following two transactions:

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

(a) Give a schedule of these two transactions that exhibits a dirty read, or explain why it is not possible.

(b) Give a schedule of these two transactions that exhibits a dirty write, or explain why it is not possible.

Solution

(a) As defined in the lectures, dirty read is not possible here, because it requires that first one transaction reads and writes a data item and then another transaction reads it, but there is no such data item in these transactions. This answer should be accepted. However, an answer where a write of a data item by one transaction is followed by a read of the same data item by another transaction and then followed by an abort of the first one, which is the essence of a dirty write, should also be accepted as correct; for example:

$$\mathtt{r}_1(Y), \mathtt{w}_1(X), \mathtt{w}_1(Z), \mathtt{w}_1(U), \mathtt{r}_2(Y), \mathtt{w}_2(X), \mathtt{r}_2(U), \mathtt{a}_1$$

(b) As defined in the lectures, dirty write is not possible here, because it requires that both transactions read and write the same data item, but there is no such data item in these transactions. This answer should be accepted. However, an answer where a write of a data item by one transaction is in between a read and a write of the same data item in another transaction, which is the essence of a dirty write, should also be accepted as correct; for example:

$$\mathtt{r}_2(Y), \mathtt{w}_2(X), \mathtt{r}_2(U), \mathtt{r}_1(Y), \mathtt{w}_1(X), \mathtt{w}_1(Z), \mathtt{w}_1(U), \mathtt{r}_1(Y), \mathtt{w}_2(U)$$

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 9 of 12 (9%)

Is it possible for a schedule to follow the two-phase locking concurrency protocol, but not follow the timestamp-based protocol? Justify your answer—that is, either give an example of such a schedule or explain why no such schedule exists.

Solution

Yes. For example, $\mathtt{b}_1, \mathtt{b}_2, \mathtt{l}_2(\mathtt{X}), \mathtt{w}_2(\mathtt{X}), \mathtt{u}_2(\mathtt{X}), \mathtt{l}_1(\mathtt{X}), \mathtt{r}_1(\mathtt{X}), \mathtt{u}_1(\mathtt{X})$

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 10 of 12 (8%)

Assume that we use the deferred update protocol. Consider a database where, before the relevant transactions start, items X and Y have values 3 and 5 respectively. Consider also the following (partial) schedule executed on this database:

$$\mathtt{b}_2, \mathtt{b}_1, \mathtt{r}_1(X), \mathtt{eval}_1(Y{:=}X*2), \mathtt{r}_2(Y), \mathtt{eval}_2(Y{:=}Y+1), \mathtt{w}_2(Y), \\ \mathtt{w}_1(Y), \mathtt{a}_1, \mathtt{b}_3, \mathtt{r}_3(Y), \mathtt{eval}_3(Y{:=}Y+1), \mathtt{w}_3(Y), \mathtt{c}_2$$

In this schedule, $\mathtt{eval}_1$, $\mathtt{eval}_2$, $\mathtt{eval}_3$ are update operations that are evaluated within transactions T_1, T_2, T_3, respectively. Assume that the last (rigid) checkpoint occurs at the beginning of the schedule, and a system crash happens at the end of the schedule. Assuming that the log buffer flushes immediately after each log update, give the log state at the moment of the crash and discuss what the system must do to recover after the crash.

Solution

Log contains the following entries (only REDO here, due to deferred update, but giving UNDO should not be penalised):

```
BEGIN T2
BEGIN T1
WRITE Y := 6 BY T2
WRITE Y := 6 BY T1
ABORT T1
BEGIN T3
WRITE Y := 7 BY T3
COMMIT T2
```

The only operation the system needs to do after the crash is to redo the operation `WRITE Y := 6 BY T2`

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 11 of 12 (8%)

In at most a few sentences, discuss the relationships between SQL and CRUD. In particular, explain whether SQL supports all CRUD operations and whether it allows data manipulation beyond CRUD.

Solution

CRUD stands for "Create, Read, Update, Delete". All these operations are realised by SQL. However, SQL allows for much more, most importantly, richer queries such as aggregates and joins.

IN3020 – Database Systems Exam, 2026

29 May, 9:00am–1:00pm (Inspera)

Question 12 of 12 (9%)

Consider the property graph created by the following Cypher clause:

```
CREATE (p1:Person {name:"Alice", born:1990})
-[m:MarriedTo {from:2015, to:2020}]-
(p2:Person {name:"Bob", born:1988})
```

Write an RDF graph that represents the same data. You may assume existence of all relevant IRIs with prefix `http://mygraph.org/`—that is, you can start your graph with `@base http://mygraph.org/` and then use shortcuts such as `<#Person>`.

Note: your knowledge of the Turtle syntax will not be evaluated, but it should be clear what triples in your RDF graph are.

Solution

One possibility:

```
@base http://mygraph.org/

<#Alice> rdf:type <#Person>.

<#Alice> <#name> "Alice" .

<#Alice> <#born> 1990 .

<#Bob> rdf:type <#Person> .

<#Bob> <#name> "Bob" .

<#Bob> <#born> 1988 .

_:b rdf:type rdf:Statement .

_:b rdf:subject <#Alice> .

_:b rdf:predicate <#marriedTo> .

_:b rdf:object <#Bob> .

_:b <#from> 2015 .

_:b <#to> 2020 .
```