

## INSTRUCTIONS

This is your exam. Complete it either at [exam.cs61a.org](http://exam.cs61a.org) or, if that doesn't work, by emailing course staff with your solutions before the exam deadline.

This exam is intended for the student with email address <EMAILADDRESS>. If this is not your email address, notify course staff immediately, as each exam is different. Do not distribute this exam PDF even after the exam ends, as some students may be taking the exam in a different time zone.

For questions with **circular bubbles**, you should select exactly *one* choice.

- ☐ You must choose either this option
- ☐ Or this one, but not both!

For questions with **square checkboxes**, you may select *multiple* choices.

- ☐ You could select this choice.
- ☐ You could select this one too!

**You may start your exam now. Your exam is due at <DEADLINE> Pacific Time.** Go to the next page to begin.

**Preliminaries**

# 1 CS W186 Spring 2021 Midterm 2

Do not share this exam until solutions are released.

**1.0.1 Contents:**

- The midterm has 5 questions, each with multiple parts, and worth a total of 100 points.

**1.0.2 Taking the exam:**

- You have 110 minutes to complete the midterm.
- You may print this exam to work on it.
- For each question, submit only your final answer on Examtool.
- For numerical answers, do not input any suffixes (i.e. if your answer is 5 I/Os, only input 5 and not 5 I/Os or 5 IOs).
- Make sure to save your answers in Examtool at the end of the exam, although the website should autosave your answers as well.

**1.0.3 Aids:**

- You may use two pages (double sided) of handwritten notes as well as a calculator.
- You must work individually on this exam.

**1.0.4 Grading Notes:**

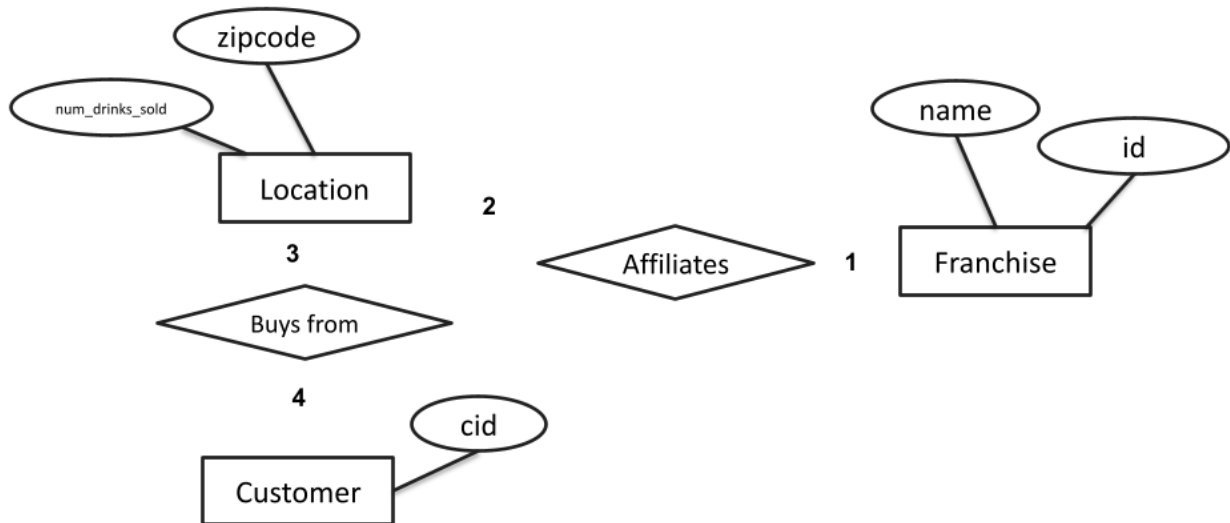
- All I/Os must be written as integers. There is no such thing as 1.02 I/Os – that is actually 2 I/Os.
- 1 KB = 1024 bytes. We will be using powers of 2, not powers of 10
- Unsimplified answers, like those left in log format, will receive a point penalty.

(a) What is your full name?

(b) What is your student ID number?

**1. (6 points) Get your boba!**

We want to model boba franchises, their affiliated locations, and customers they have sold boba to. Consider the ER diagram below.



Assumptions:

- Each franchise's **id** is a unique identifier.
- Each location is affiliated with exactly one franchise, each franchise must have at least 1 location.
- Each franchise has at most one location in a given zipcode, but multiple franchises can have locations in the same zipcode.
- Customers can buy boba from multiple locations and can buy any number of drinks, including no drinks.

(a) (1 pt) What type of edge should be drawn at 1 from Franchise to Affiliates?

- ☐ Thin line  
☒ Bold line  
☐ Thin arrow  
☐ Bold arrow

(b) (1 pt) What type of edge should be drawn at 2 from Location to Affiliates?

- ☐ Thin line  
☐ Bold line  
☐ Thin arrow  
☒ Bold arrow

(c) (1 pt) What type of edge should be drawn at 3 from Location to Buys from?

- ☒ Thin line
- ☐ Bold line
- ☐ Thin arrow
- ☐ Bold arrow

(d) (1 pt) What type of edge should be drawn at 4 from Customer to Buys from?

- ☒ Thin line
- ☐ Bold line
- ☐ Thin arrow
- ☐ Bold arrow

(e) (2 pt) What attribute(s) should be part of the Location relation's primary key? If there are multiple attributes, separate them with commas.

zipcode, Franchise.id

We store affiliates as part of Location as it is a weak entity.

**2. (14 points) Functional What?****(a) (8 points)**

Consider the relation  $R = ABCDE$ .

Determine if each of the following sets of functional dependencies will lead to  $R$  having *exactly two* different candidate keys (doesn't need to contain the same number of attributes). If so, write "Yes" below and write out what those candidate keys are. For instance if the keys are **AB** and **C**, then write "Yes. AB, C". Otherwise, write "No". No explanations needed.

i. (2 pt)  $A \rightarrow B, B \rightarrow C$

No

ii. (2 pt)  $A \rightarrow BC, B \rightarrow D, C \rightarrow DE$

No

iii. (2 pt)  $AB \rightarrow DE, B \rightarrow C, D \rightarrow E, CD \rightarrow A$

Yes. AB, BD

iv. (2 pt)  $B \rightarrow CDE, C \rightarrow B, D \rightarrow A$

Yes. B, C

**(b) (6 points)**

Consider the relation  $R = ABCDE$ .

For the following set of functional dependencies, determine if it will lead to R in BCNF without any further decomposition. If so, write “Yes” below. Otherwise, write “No” and the functional dependency that is violated. For instance if it violates the dependency  $A \rightarrow B$ , then write “No.  $A \rightarrow B$ ”. You only need to write one of the violations even if multiple exists.

- i. (2 pt)  $A \rightarrow BC$ ,  $B \rightarrow CD$ ,  $C \rightarrow DE$ ,  $D \rightarrow EA$

Yes

- ii. (2 pt)  $B \rightarrow CDE$ ,  $C \rightarrow B$ ,  $D \rightarrow A$

No.  $D \rightarrow A$

- iii. (2 pt)  $AB \rightarrow CD$ ,  $B \rightarrow CDE$ ,  $C \rightarrow DE$ ,  $CE \rightarrow AB$

Yes

**3. (35 points) There's no Transaction Fees!****(a) (6 points) True or False**

For the following 6 questions, indicate whether the statement is true or false.

- i. (1 pt) For a given schedule, it is possible for the same transaction to abort under both wait-die and wound-wait deadlock prevention policies.

☒ True

☐ False

Consider the following schedule: T1 - X(A), T3 - S(B), T3 - S(A), T2 - X(B) with priorities  $1 > 2 > 3$ . T3 will abort under wait-die when it tries to acquire S(A), since T1 has higher priority than T3. T3 will also abort under wound-wait when T2 tries to acquire X(B), since T2 has higher priority than T3.

- ii. (1 pt) Guaranteeing view serializability is sufficient for guaranteeing isolation from the ACID properties.

☒ True

☐ False

View serializability ensures serializability, which is how isolation is enforced.

- iii. (1 pt) The waits-for graph is the conflict dependency graph with edges reversed.

☐ True

☒ False

The dependency graph does not consider which locks each transaction holds at each timestep, whereas the waits-for graph does.

- iv. (1 pt) A view serializable schedule will always satisfy strict 2PL requirements.

☐ True

☒ False

A read-only schedule will satisfy view serializability, but it may still violate strict 2PL if all locks are not released together.

- v. (1 pt) During deadlock no transactions in the database will make progress.

☐ True

☒ False

Transactions not involved in deadlock can still proceed.

- vi. (1 pt) Multiple locking granularity provides no throughput benefit for a workload consisting entirely of full table scans compared to table-only locking.

☒ True

☐ False

Table scans consist simply of read-only operations, which would not block each other. Multiple locking granularity would in fact reduce throughput by adding additional overhead.

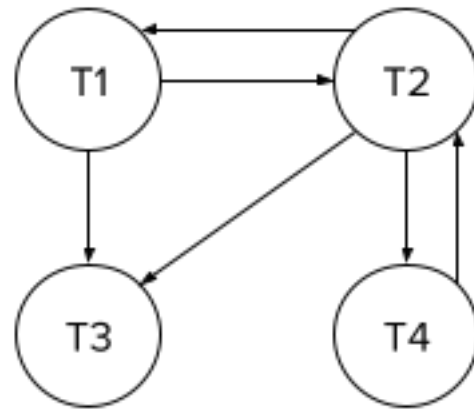
**(b) (3 points) Conflict Dependencies**

Consider the following schedule. The top row is the timestep of each operation.

Transaction	1	2	3	4	5	6	7	8	9	10
T1	R(A)					W(C)				
T2		W(B)			R(C)		W(C)		W(D)	
T3			W(A)					R(B)		
T4				R(D)						W(D)

i. (2 pt) How many edges in the conflict dependency graph of the above schedule point to T2?

2



See the conflict dependency graph.

ii. (1 pt) Select all of the following serial schedules that are conflict equivalent to the above schedule.

☐ T1, T2, T3, T4

☐ T1, T4, T3, T2

☐ T4, T2, T3, T1

☐ T3, T2, T4, T1

☒ None of the above

The schedule is not conflict serializable, so it will not be conflict equivalent to any serial schedule.



**(c) (9 points) Deadlock Detection and Prevention**

Consider the following schedule. The top row is the timestep of each operation.

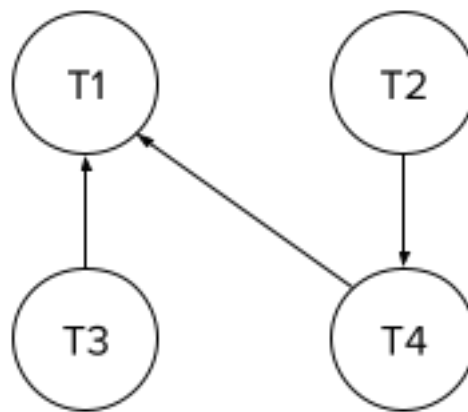
This schedule consists of multi granularity locks as presented in lecture. The resources are: Tables *A* and *B* with pages *A1*, *A2* and *B1*, *B2* respectively. Assume every transaction has an IX lock on the database.

There is queue skipping for this problem. Queue skipping allows a transaction to acquire any compatible locks, even if the wait queue for the resource is not empty.

Transaction	1	2	3	4	5	6	7	8	9	10	11
T1	IS(A)						S(A1)				
T2		IS(B)				S(A)			S(B2)		
T3			X(A)					SIX(B)			X(B1)
T4				IX(B)	SIX(A)					X(A1)	

- i. (2 pt) Select all of the following transactions that point towards T1 in the waits-for graph for the schedule above.

- ☐ T2  
☒ T3  
☒ T4  
☐ None of the above



See the waits-for graph.

- ii. (1 pt) Select all of the following transactions that are involved in deadlock for the schedule above.

- ☐ T1  
☐ T2  
☐ T3  
☐ T4  
☒ No deadlock

See the waits-for graph in the previous solution. There are no cycles so there is no deadlock.

- iii. (3 pt) Select all of the following transactions that will abort under the **wait-die** deadlock prevention policy for the schedule above.

The priorities in *descending* order are: T1, T2, T3, T4 (i.e. T1 has the greatest priority and T4 has the least priority).

☐ T1

☐ T2

☒ T3

☒ T4

☐ None of the above

- T3 aborts at TS (timestep) 3, because X(A) conflicts with IS(A) and T3 has lower priority than T1
- T2 waits for T4 at TS 6, because S(A) conflicts with SIX(A), but S(A) does not conflict with IS(A). T2 has higher priority than T4, so T2 waits.
- T4 aborts at TS 10, because X(A1) conflicts with S(A1) and T4 has lower priority than T1.

- iv. (3 pt) Select all of the following transactions that will abort under the **wound-wait** deadlock prevention policy for the schedule above.

The priorities are the same as the previous question.

☐ T1

☐ T2

☐ T3

☒ T4

☐ None of the above

- T3 waits at TS 3, because X(A) conflicts with IS(A) and T3 has lower priority than T1.
- T4 aborts at TS 6, because S(A) conflicts with SIX(A) and T2 has higher priority than T4.

**(d) (13 points) Serializability**

Consider the following incomplete Java code that checks whether or not a schedule is serializable.

```

/* Returns whether or not the Schedule S is serializable */
public static boolean is_serializable(Schedule s) {
    /* BEGIN BLOCK A */
    if (____(1)____) {
        return true;
    }
    if (is_view_serializable(s)) {
        ____ (2) ____;
    }
    /* END BLOCK A */

    for (Schedule ss: serial_schedules(s)) {
        if (____(3)____) {
            return true;
        }
    }
    return false;
}

/* Returns whether or not the Schedule S is conflict serializable */
public static boolean is_conflict_serializable(Schedule s) {
    Graph graph = new Graph(s.transaction_ids);
    Edges edges = ____ (4) ____;
    graph.add_edges(edges);
    return ____ (5) ____;
}

```

The functions shown below may be helpful to complete the blanks above. Implementations are not shown on purpose.

```

/* Returns whether or not the Schedule S is view serializable */
public static boolean is_view_serializable(Schedule s) {...}

/* Returns a List of all possible Schedules if S were serial */
public static List<Schedule> serial_schedules(Schedule s) {...}

/* Returns whether or not Schedules S1 and S2 are equivalent */
public static boolean equivalent(Schedule s1, Schedule s2) {...}

public class Schedule {
    // List of transaction IDs
    public List<Integer> transaction_ids;
    // Conflicting operations where 1st is a read and 2nd is a write
    public Edges rw_edges() {...}
    // Conflicting operations where 1st is a write and 2nd is a read
    public Edges wr_edges() {...}
    // Conflicting operations where 1st is a write and 2nd is a write
    public Edges ww_edges() {...}
}

public class Graph {
    public Graph(List<Integer> nodes) {...} // Creates Graph with NODES as nodes
    public void add_edges(Edges edges) {...} // Adds EDGES to this graph
}

```

```

        public boolean is_cyclic() {...}           // Checks whether graph has cycle
    }

    // A collection of edges in a graph.
    public class Edges {
        // Result of concatenating this Edges object with OTHER
        public Edges concat(Edges other) {...}
    }

```

The next 5 questions involve filling in the blanks above.

**Exact Java syntax is not necessary to get full credit, but try to be as accurate as possible.**

- i. (2 pt) What belongs in the blank labeled (1)?

```
is_conflict_serializable(s)
```

- ii. (2 pt) What belongs in the blank labeled (2)?

```
return true
```

- iii. (2 pt) What belongs in the blank labeled (3)?

```
equivalent(s, ss)
```

- iv. (3 pt) What belongs in the blank labeled (4)?

```
s.rw_edges().concat(s.wr_edges()).concat(s.wv_edges())
```

- v. (3 pt) What belongs in the blank labeled (5)?

```
!graph.is_cyclic()
```

- vi. (1 pt) Consider a change to the function `is_serializable` above, where the code within BLOCK A is removed. Would this modified code still correctly check for serializability?

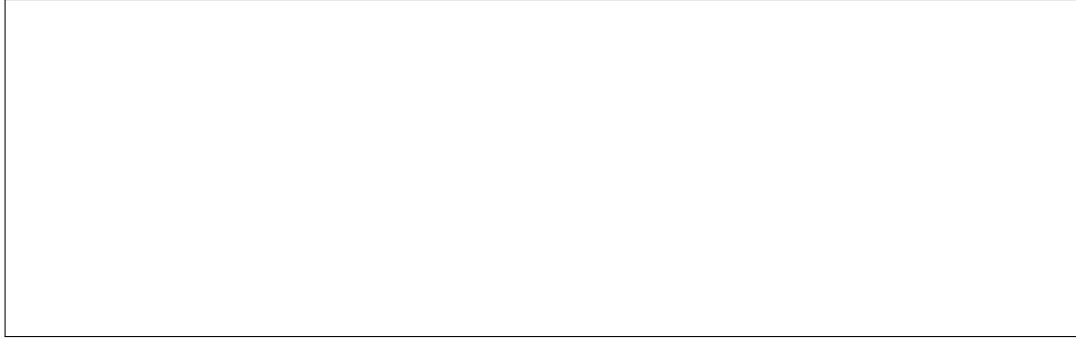
- ☒ Yes  
☐ No

Yes, it would still be correct, because we are just removing checks for conflict and view serializability. Any conflict or view serializable schedule is still serializable, and the serializability check is handled within the `for` loop. The function's best case will now be slower, because conflict serializability is cheaper to check than serializability.

**(e) (4 points) Design Problem**

- i. **(4 pt)** Consider a change where a transaction must release all of its held locks when it is placed on the wait queue for any lock. It must then re-acquire any necessary locks it had previously released upon leaving the wait queue.

In no more than 4 sentences explain whether or not this change prevents all deadlock.



Yes, this prevents deadlocks, because no transaction will wait for a transaction that is in a wait queue. Transactions may however take a lot longer to complete and isolation properties are easily violated.

**4. (21 points) Que-rum Op-tea-miza-sugar**

Consider the following tables:

```
CREATE TABLE T1 (
  x INTEGER,
  y FLOAT,
  z FLOAT
);
```

```
CREATE TABLE T2 (
  x INTEGER,
  y FLOAT,
  z FLOAT
);
```

```
CREATE TABLE T3 (
  x INTEGER,
  y FLOAT,
  z FLOAT
);
```

Number of pages and records/page (assume all pages are full of records):

T1: 100 pages, 50 records/page,

T2: 200 pages, 50 records/page,

T3: 150 pages, 50 records/page

Indexes:

T1: Alt 1 index on T1.x with h = 2 and 120 leaf pages

T2: Alt 2 unclustered index on T2.y with h = 3 and 50 leaf pages

Table Stats (assume all are uniformly distributed and min/max are inclusive):

T1.x: min = 0, max = 50, 51 unique values

T2.y: min = 0, max = 100, 500 unique values

T2.z: min = 0, max = 200, 200 unique values

T3.y: min = 50, max = 150, 250 unique values

Buffer pages

B = 12

**(a) (4 points) Selectivity Exercises**

Estimate the number of pages of output for each of the following queries.

i. (1 pt) `SELECT * FROM T1 WHERE x < 40;`

**79**

Selectivity:  $(40 - 0) / (50 - 0 + 1) = 40/51$

Number of tuples:  $\text{floor}(40/51 * 100 * 50) = 3921$

Number of pages:  $\text{ceil}(3921 / 50) = 79$

- ii. (1 pt) SELECT \* FROM T2 WHERE y >= 30 and y <= 90;

120

Selectivity:  $(90 - 30) / 100 = 0.6$

Number of tuples:  $\text{floor}(200 * 50 * 0.6) = 6000$

Number of pages:  $\text{ceil}(6000 / 50) = 120$

- iii. (2 pt) SELECT T2.x, T2.y, T2.z FROM T2, T3 WHERE T2.y = T3.y AND T2.z >= 20;

2700

Selectivity:  $1 / \max(\text{num distinct T2.y, num distinct T3.y}) * (\max(\text{T2.z}) - 20) / (\max(\text{T2.z}) - \min(\text{T2.z}))$

Selectivity:  $1 / 500 * (200 - 20) / 200 = 0.0018$

Number of tuples:  $[\text{T2}] * \text{T2 records/page} * [\text{T3}] * \text{T3 records/page}$

Number of tuples (cartesian product):  $200 * 50 * 150 * 50 = 75,000,000$

Number of tuples (final):  $\text{floor}(75,000,000 * 0.0018) = 135000$

Number of pages (final):  $\text{ceil}(135000 / 50) = 2700$

**(b) (17 points) Query Optimizers**

Again consider the following tables:

```
CREATE TABLE T1 (  
  x INTEGER,  
  y FLOAT,  
  z FLOAT  
);
```

```
CREATE TABLE T2 (  
  x INTEGER,  
  y FLOAT,  
  z FLOAT  
);
```

```
CREATE TABLE T3 (  
  x INTEGER,  
  y FLOAT,  
  z FLOAT  
);
```

Number of pages and records/page:

T1: 100 pages, 50 records/page,

T2: 200 pages, 50 records/page,

T3: 150 pages, 50 records/page

Indexes:

T1: Alt 1 index on T1.x with h = 2 and 120 leaf pages

T2: Alt 2 unclustered index on T2.y with h = 3 and 50 leaf pages

Table Stats (assume all are uniformly distributed):

T1.x: min = 0, max = 50, 50 unique values

T2.y: min = 0, max = 100, 500 unique values

T2.z: min = 0, max = 200, 200 unique values

T3.y: min = 50, max = 150, 250 unique values

Buffer pages

B = 12

We want to execute the following query:

```
SELECT *  
  FROM T1 INNER JOIN T2 ON T1.x = T2.x  
        INNER JOIN T3 ON T2.y = T3.y  
 WHERE T2.y >= 20  
 ORDER BY T2.y;
```

i. (1 pt) How many I/Os would a full scan of T1 take?

100

Full scan is all 100 pages.



- ii. (2 pt) How many I/Os would an index scan of T1 using the index on T1.x take?

122 or 120 (leaving out inner nodes)

2 inner nodes + 120 leaf pages

- iii. (2 pt) Using the index on T2.y, how many I/Os would it take to perform an index scan of T2 that only returns tuples which match the condition  $T2.y \geq 20$ ?

8043

$\text{Sel}(T2.y \geq 20) = 0.8$

3 inner nodes

$0.8 * 50 = 40$  leaf nodes

$0.8 * 200 * 50 = 8000$  records (1 I/O per matching record)

$3 + 40 + 8000 = 8043$

- iv. (2 pt) Assume in the first pass of the Selinger Optimizer that T2 and T3 were both accessed with a full scan and streamed directly into the second pass. How many I/Os then would T3 BNLJ T2 (T3 as outer relation) take, including the cost of the 1st pass? Remember that the number of buffer pages B = 12.

3150

$[T3] + ([T3] / (B - 2)) * [T2]$

$150 + (150 / 10) * 200 = 3150$

- v. (3 pt) Now, assume the full scan on T2 filtered  $T2.y \geq 20$  and was materialized before the same T3 BNLJ T2 join. How many I/Os would the join take with this one modification, again including the cost of the 1st pass?

2910

$\text{Sel}(T2.y \geq 20) = 0.8$

$[T2] \text{ (read T2)} + 0.8[T2] \text{ (materialize T2)} + [T3] + ([T3] / (B - 2)) * 0.8[T2]$

$200 + 0.8(200) + 150 + (150 / 10) * 0.8(200) = 2910$

vi. (3 pt) Here is the query plan again:

```
SELECT *
  FROM T1 INNER JOIN T2 ON T1.x = T2.x
        INNER JOIN T3 ON T2.y = T3.y
 WHERE T2.y >= 20
 ORDER BY T2.y;
```

Which of the following query plans will be returned by the second pass of the Selinger Optimizer given total estimated I/O cost and output order?

A:	T1 BNLJ T2	I/Os: 3,000	Order: None
B:	T2 BNLJ T1	I/Os: 2,000	Order: None
C:	T1 SMJ T2	I/Os: 5,000	Order: T1.x
D:	T1 BNLJ T3	I/Os: 6,000	Order: None
E:	T3 BNLJ T1	I/Os: 2,000	Order: None
F:	T3 SMJ T1	I/Os: 1,500	Order: T3.z
G:	T2 BNLJ T3	I/Os: 4,000	Order: None
H:	T3 BNLJ T2	I/Os: 2,500	Order: None
I:	T2 SMJ T3	I/Os: 3,500	Order: T2.y

☐ A

☒ B

☐ C

☐ D

☐ E

☐ F

☐ G

☒ H

☒ I

Cheapest T1,T2 and T2,T3 joins are kept. T2 SMJ T3 is kept since it has an interesting order.

vii. (3 pt) Consider the following modification to the Selinger Optimizer: after we determine our **final** query plan, we test if materializing the output of each operator (single access or join) before inputting it to the next operator will reduce the I/O cost. For our query above, how many query plans would we have to consider after the final step of the Selinger Optimizer if we consider all possible combinations of materialization/no materialization?

32

There are 5 operators for this query (3 single access, 2 joins) so in total there are  $2^5 = 32$  combinations of materialization/no materialization.

**viii. (1 pt)** True or False? Pushing down on-the-fly selections in single access scans will always reduce the final I/O cost of the query plan outputted by the Selinger Optimizer.

☐ True

☒ False

Counterexample: Pushing down the selection on the right relation of a BNLJ will not decrease I/O cost.

[Discussion 7 example here](#)

Also, more simple: Consider a predicate that evaluates to true for all tuples.

**5. (24 points) Join me in joining some tables!**

For all parts, assume that table R has 60 pages and table S has 40 pages. Exclude the cost of the final write unless instructed otherwise.

**(a) (2 points) What's your favorite join?**

Dylan wants to join table R with table S using Block Nested Loop Join. For this part only, assume we have 15 buffers and answer the following questions below.

**i. (1 pt)** What is the I/O cost of R BNLJ S (R is the outer relation)?

**260**

$$60 + 5 * 40 = 260 \text{ I/Os}$$

**ii. (1 pt)** What is the I/O cost of S BNLJ R (S is the outer relation)?

**280**

$$40 + 4 * 60 = 280 \text{ I/Os}$$

**(b) (6 points) Block Nested Lakshya Join (BNLJ)**

$[R] = 60$ ,  $[S] = 40$

Lakshya also wants to join table R with table S. He has 20 pages in his buffer, but there's a catch... each time a block from the outer relation is evicted and a new block is read in, the available memory shrinks by 5 pages until we hit a floor of  $B = 5$  total buffer pages.

In other words, when the first block was read in there were 20 pages in the buffer. When the next block was read in, there were only  $20 - 5 = 15$  pages in the buffer.

- i. (3 pt) What is the I/O cost for executing BNLJ here, if R was the outer relation?

460

- [1] 18 pg block + 40 pages
- [2] 13 pg block + 40 pages
- [3] 8 pg block + 40 pages
- [4] 3 pg block + 40 pages
- [5] 3 pg block + 40 pages
- [6] 3 pg block + 40 pages
- [7] 3 pg block + 40 pages
- [8] 3 pg block + 40 pages
- [9] 3 pg block + 40 pages
- [10] 3 pg block + 40 pages

- ii. (3 pt) If instead, the available memory was to expand by 5 pages (starting at  $B = 20$ ) every time a block from the outer relation is evicted and a new one read in, what would be the cost for BNLJ, if R was the outer relation?

180

- [1] 18 pg block + 40 pages
- [2] 23 pg block + 40 pages
- [3] 19 pg block + 40 pages

**(c) (10 points)    Sorting things out**

$$[R] = 60, [S] = 40$$

A new resource management model means available buffer pages can change throughout query execution. For each pass in the sort phase of SMJ, Lakshya has fewer buffer pages than the previous pass! The allocation is detailed in the figures below.

Lakshya wants to use SMJ to join his tables ( $R \text{ SMJ } S$ ). Consider the following figure for the number of buffer pages available for each pass and answer the questions below. **Assume no optimization is used.**

SMJ	Pass 0	Pass 1	Pass 2
	10	5	?

- i. (1 pt) What is the I/O cost for pass 0 of sorting table R?

120

$$60 * 2 = 120$$

- ii. (1 pt) What is the I/O cost for pass 0 of sorting table S?

80

$$40 * 2 = 80$$

- iii. (1 pt) How many sorted runs do we have at the end of pass 0 for table R?

6

$$\text{ceil}(60/10) = 6$$

- iv. (1 pt) How many sorted runs do we have at the end of pass 0 for table S?

4

$$\text{ceil}(40/10) = 4$$

- v. (1 pt) How many pages are in our smallest run from R at the end of Pass 1?

20

We first merge 4 runs of 10 pages each and get 1 run of 40 pages. We then merge the remaining 2 runs of 10 pages each to get 1 run of 20 pages.

- vi. (1 pt) What's the minimum number of buffer pages we must have in order to finish sorting after pass 2?

3

We have 2 runs of table R after pass 1 (above solution has breakdown)

vii. (1 pt) What is the average cost of the merge pass of R SMJ S?

100

$$[R] + [S] = 100$$

viii. (3 pt) Suppose we have the following records in table R and table S

Table R	Table S
1	1
1	2
2	2
3	3
3	4

How many times will we have to backtrack when performing the merge pass of SMJ (i.e. resetting the right iterator over table S)?

4

4 or 5 (depending on when you reset; either is accepted)

## (d) (6 points) Direct Deposit: \$1400, Me: give me all the buffer pages

[R] = 60, [S] = 40

Lakshya wants to try using GHJ. Like before, for each pass of GHJ, Lakshya has fewer buffer pages than the previous pass. Consider the following figure for the number of buffer pages available for each pass and answer the questions below. Assume that our pass 1 hash function hashes  $\frac{1}{2}$  of the pages to the first bucket,  $\frac{1}{4}$  pages to the second bucket, and perfectly hashes the remaining pages uniformly across the remaining buckets,

GHJ	Partitioning Pass 1	Partitioning Pass 2	Build and Probe
	20	?	12

## i. (3 pt) What is the total I/O cost of pass 1?

209

Table R: 60 (read) + 30 (bucket 1) + 15 (bucket 2) + 1 \* 17 (buckets 3 to 19 combined) = 60 + 62 = 122 I/Os

Table S: 40 (read) + 20 (bucket 1) + 10 (bucket 2) + 1 \* 17 (buckets 3 to 19) = 40 + 47 = 87 I/Os

Table R + Table S = 122 + 87 = 209 I/Os

Partial: 109 (did not include first read)

ii. (3 pt) Assuming we use a uniform hash function in pass 2, what is the minimum number of buffer pages that must be available in pass 2 in order to ensure that this is the **last partitioning phase**?

3

To fit everything in the build and probe we only need to ensure that all partitions have one table of size 10 pages or less ( $B - 2 = 10$ ) – this enables us to build an in-memory hash table. As we use a uniform hash function in pass 2, it is sufficient to ensure that the largest bucket can be hashed into subpartitions – we just need to hash S into a small enough partition to build an in-memory table on it and then probe with R.



**No more questions.**