

# CS 186/286 Spring 2018 Midterm 2

- Do not turn this page until instructed to start the exam.
- You should receive 1 single-page *answer sheet* and a 11-page *exam packet*.
- All answers should be written on the answer sheet. The exam packet will be collected but not graded.
- You have *80 minutes* to complete the midterm.
- The midterm has *4 questions*, each with multiple parts.
- The midterm is worth a total of *60 points*.
- For each question, place only your *final answer* on the answer sheet; do not show work.
- For multiple choice questions, please fill in the bubble or box completely, **do not mark the box with an X or checkmark**.
- Use the blank spaces in your exam for scratch paper.
- You are allowed **one** 8.5" × 11" double-sided page of notes.
- No electronic devices are allowed.

# 1 Joins (20 points)

Note: Assume that when we write  $X \bowtie Y$ ,  $X$  (on the left) is the outer, and  $Y$  (on the right) is the inner for the remainder of these questions.

- (1 point) Suppose we do a Page-oriented Nested Loop Join on two tables,  $P$  and  $Q$ . We are told that the same amount of pages are required to store  $P$  and  $Q$ , but  $Q$  has three times more records per page than  $P$ , on average. Which one of the following options is true:
  - $P \bowtie Q$  does fewer I/Os than  $Q \bowtie P$ .
  - $Q \bowtie P$  does fewer I/Os than  $P \bowtie Q$ .
  - $P \bowtie Q$  does the same number of I/Os as  $Q \bowtie P$ .
- (5 points) Next, we consider a Simple Nested Loop Join on three tables:  $P$ ,  $Q$ , and  $S$ .
  - Please use  $[X]$  to denote the number of pages of  $X$ , and  $\{X\}$  to denote the number of records in  $X$  (note the curly braces rather than vertical bars, for legibility).
  - Do not materialize intermediary results into a temporary table. Assume the results of a subplan are pipelined to the next operator.
  - Assume  $P \bowtie Q$  is foreign key to primary key join, and the following join with  $S$  is also a foreign key to primary key join. The (non null) foreign keys are in  $P$ , and primary keys are in  $Q$  and  $S$ .
  - (1 point) What is the I/O cost for  $P \bowtie Q$ ?
  - (1 point) What is the cardinality of  $P \bowtie Q$ ?
  - (2 points) What is the I/O cost for  $(P \bowtie Q) \bowtie S$ ?
  - (1 point) What is the cardinality of  $(P \bowtie Q) \bowtie S$ ?
- (1 point) Which one of the following options is true about the IO cost of a self join (i.e. joining a table with itself)? Assume  $B \gg 3$  where  $B$  is the pages of memory available.
  - $\text{IO}(\text{SNLJ}) > \text{IO}(\text{PNLJ}) \geq \text{IO}(\text{BNLJ})$
  - $\text{IO}(\text{SNLJ}) \leq \text{IO}(\text{PNLJ}) \leq \text{IO}(\text{BNLJ})$
  - $\text{IO}(\text{SNLJ}) \geq \text{IO}(\text{PNLJ}) \geq \text{IO}(\text{BNLJ})$
  - $\text{IO}(\text{PNLJ}) \leq \text{IO}(\text{SNLJ}) \leq \text{IO}(\text{BNLJ})$
- (3 points) Suppose there are two tables:  $R$  with 40 pages and  $S$  with 75 pages. What is the cost of sort-merge join between tables  $R$  and  $S$ ? Assume the operator has 6 pages of RAM, and it is not writing the output to disk. Use the optimization from lecture if possible.
- (10 points) Suppose there are two tables:  $P$  and  $Q$ . Table  $P$  has 1,200 pages and 35 records per page. Table  $Q$  has 100 pages and 50 records per page. In this question, we consider an equality join  $P \bowtie Q$ . Assume that join key values are widely and uniformly distributed. That is, there is no skew. Also assume that although the join operator allocates 1 buffer for the output, it does not write the output to disk.
  - (1 point) What is the I/O cost of a grace hash join? Assume the join has 11 buffer pages of memory available.
  - (1 point) If we use a hybrid hash join, how many pages will be allocated for the hash table in the first phase assuming a fill factor of 0.9? Also assume the join has 32 buffer pages of memory available.
  - (3 points) How many I/Os will the first pass of the hybrid hash join incur?

- (d) (3 points) How many I/Os will the second pass of the hybrid hash join incur?
- (e) (2 points) Suppose we have an index on  $Q$  with a compound key whose prefix is the join key. It is an alternative 2 clustered index with 40 leaf pages total. Assume that the upper levels of the tree are cached with a few spare (extra) buffer pages in the cache. Also assume that the query needs no additional columns from  $Q$ . Assume that the join key is a primary key in  $Q$ .  
How many I/Os does INLJ take?

## 2 Parallel Queries (8 points)

The Jedi Order decides to create a database tracking all the Fighter Pilots, Sith Lords, and Jedi Masters throughout history. The order creates the following relations:

- **Jedi** (name, planet, padawan)
- **Sith** (name, planet, apprentice)
- **Pilot**(name, droid, plane, num\_fighters\_destroyed)

And they are partitioned across 3 machines, each with  $B$  pages of memory in the following manner (note that the Jedi Order –mistakenly–assumes a uniform distribution for all values when partitioning):

- **Jedi**: Hash-partitioned on padawan over machines 1, 2, 3.
  - **Sith**: Hash-partitioned on apprentice over machines 1, 2, 3.
  - **Pilot**: Range-partitioned on num\_fighters\_destroyed over machines 1, 2, 3.
1. (1 point) The Jedi Order realizes its pilots are woefully inexperienced - 80% of them have num\_fighters\_destroyed = 0, because they have never had any combat experience! Let us say that we wish to do a parallel scan on this table. Let  $[P]$  denote the number of pages in **Pilot** and let  $T$  denote the time taken for one I/O. How long will it take for a parallel scan to complete?
    - A.  $1/3 * [P] * T$
    - B.  $[P] * T$
    - C.  $0.8 * [P] * T$
    - D.  $0.2 * [P] * T$

The order knows that many of the Sith used to be Jedi, and so they decide to perform a hash-based join on the **Jedi** and **Sith** tables to find who exactly is both Jedi and Sith. Assume that we join **only** on the name field, that we are using perfect hash functions and that the distribution of name values is not skewed for both **Jedi** and **Sith**. Let  $[J]$  be the number of pages in **Jedi** and  $[S]$  be the number of pages in **Sith**, with  $[S] < [J]$ .

2. (1.5 points) Suppose we first wanted to remove duplicate entries in **Sith** using hashing. What is the number of passes required to re-partition **Sith** across all 3 machines and de-duplicate the Sith records? Include the shuffle phase in your calculations. Assume we are using the *optimized* variant of parallel hashing in which shuffling the records is pipelined into the hashing operator. Assume the operator has  $B$  buffer pages of available memory.
  - A.  $2 + \lceil \log_{B-1}(\lceil \frac{[S]}{B} \rceil) \rceil$
  - B.  $2 + \lceil \log_{B-1}(\lceil \frac{[S]}{3B} \rceil) \rceil$
  - C.  $1 + \lceil \log_{B-1}(\lceil \frac{[S]}{B} \rceil) \rceil$
  - D.  $1 + \lceil \log_{B-1}(\lceil \frac{[S]}{3B} \rceil) \rceil$

Now assume that the order introduces a new table called **padawans**, which tracks apprentice Jedi who aspire to (but are not yet) be Knights. Each Padawan has one master who trains them. The schema for padawans is as follows:

**padawans** (name, age, master)

Assume this table is round-robin partitioned across the 3 machines. Now, assume that 60% of all padawans in the table have "Yoda" as their master. We wish to run the following query on the table:

```
SELECT * FROM padawans WHERE master = "Yoda"
```

3. (1 point) What is the time taken for the above query to complete, if each machine has  $B$  buffer pages and `padawans` has  $[P]$  pages in total? Let  $T$  denote the time taken for 1 I/O.

- A.  $\frac{1}{3} * [P] * T$
- B.  $0.6 * [P] * T$
- C.  $[P] * T$
- D.  $0.6 * \frac{[P]}{B} * T$

4. (1 point) If we wanted to sort `padawans` by the `age` key, whose values are uniformly distributed, what partitioning scheme would result in the fewest I/Os?

- A. Hash Partitioning on age
- B. Hash Partitioning on name
- C. Range Partitioning on name
- D. Range Partitioning on age

5. (1.5 points) Suppose we have time series of all monitoring events from all our fighter ships for the last 2 years:

```
events(event_time, event_type, duration, ship_id, pilot_name, pilot_rating)
```

These records tell us what is going on with our fighter fleet. To keep the fleet fighting aggressively, our engineers are constantly running queries over the latest data and for a specific event – the low fuel alert. Which partitioning scheme and physical design will achieve the best average latency if we have many engineers querying concurrently ?

- A. Hash partition on `event_type` and clustered index on `event_time`
- B. Range partition on `event_time` and unclustered index on `event_type`
- C. Hash partition on (`ship_id`, `pilot_name`) and clustered index on (`event_time`, `event_type`)
- D. Range partition on `duration` and unclustered index on `event_time`

6. (2 points) True or False. Mark all of the true answers and leave all of the false ones blank.

- A. If we partition data using the round robin scheme and then perform some operations on the data, every machine is guaranteed to have the same number of records  $\pm 1$  record.
- B. Range partitioning keeps data stored in sorted order on each machine.
- C. A join over two tables that requires an asymmetric shuffle will incur fewer network I/Os than one that requires a symmetric shuffle.
- D. If data is partitioned using round robin, when searching for a record, we have to perform broadcast lookup to only a subset of the machines.

### 3 Query Optimization (20 points)

For this section, remember that **the height of a one level tree is defined to be 0, the height of a tree with two levels is defined to be 1, and so on.**

Suppose the System R assumptions about uniformity and independence from lecture hold. Assume that costs are estimated as a number of I/Os, without differentiating random and sequential I/O cost. Consider the following relational schema:

```
Item (iid, name, price, popularity)
Store (sid, zip, city)
Sales (iid, sid, quantity)
```

Consider the following query:

```
SELECT name
FROM Item
WHERE price >= 100 AND price < 200
```

Suppose we have the following statistics on table Item:

**tuples: 5000, tuples/page: 100, total pages: 50, price range: 0 – 4999**

1. (1 point) What is the estimate for the number of records output from this query?
2. (1 point) If we build an unclustered alternative 2 B+-tree index of height 2 on Item.price and use an index scan to process this query, what is the number of IOs incurred? Assume that that we only need to read at most one leaf node and assume that we can cache the index.
3. (1 point) True or False: If we build a clustered alternative 2 B+-tree index of height 2 on Item.price and use an index scan to process this query, the number of IOs is less than the number of IOs required for a sequential scan. Assume that that we only need to read at most one leaf node and assume that we can cache the index.
  - A. True
  - B. False

Consider the following query with a histogram on Sales.quantity:

```
SELECT name, price
FROM Item, Store, Sales
WHERE Item.iid = Sales.iid AND Store.sid = Sales.sid AND Sales.quantity >= 100
ORDER BY price
```

0-100	100-200	200-300	300-400	400-500
15%	20%	20%	15%	30%

Figure 1: Histogram on Sales.quantity.

4. (1 point) True or False: Assume a sale always has a corresponding item and store. Applying a filter using the histogram on Sales.quantity before joining always yields a cheaper plan for this query.
  - A. True
  - B. False
5. (1 point) Based on the histogram on Sales.quantity, what is the selectivity for Sales.quantity  $\geq 100$  ? Assume that the histogram boundaries are left inclusive and right exclusive.
6. (1 point) True or False: The reason System R optimizer will not consider joining two tables with no join predicate (i.e. performing a Cartesian product) is because these plans will always be suboptimal.
  - A. True
  - B. False
7. (2 points) Suppose we have the following list of 2 table subplans and their costs. Mark the letters of the plans that will be chosen by the optimizer. Ignore interesting orders.
  - A. Item  $\bowtie$  Sales (2,000 IO)
  - B. Sales  $\bowtie$  Item (1,500 IO)
  - C. Store  $\bowtie$  Sales (3,000 IO)
  - D. Sales  $\bowtie$  Store (4,000 IO)
8. (4 points) Based on the plans that were retained in the last pass (the previous question), now suppose we have the following list of 3 table access plans and their costs. Mark the letters for all 3 table access plans that would be considered by the optimizer and write down the letter of the final query plan that will be chosen. (Cost given are cumulative for the full query. Again, ignore interesting orders in formulating your answer.)
  - A. (Item  $\bowtie$  Sales)  $\bowtie$  Store (2,500,000 IO)
  - B. (Sales  $\bowtie$  Item)  $\bowtie$  Store (2,000,000 IO)
  - C. Store  $\bowtie$  (Item  $\bowtie$  Sales) (1,800,000 IO)
  - D. Store  $\bowtie$  (Sales  $\bowtie$  Item) (1,500,000 IO)
  - E. (Sales  $\bowtie$  Store)  $\bowtie$  Item (4,500,000 IO)
  - F. (Store  $\bowtie$  Sales)  $\bowtie$  Item (3,500,000 IO)
  - G. Item  $\bowtie$  (Sales  $\bowtie$  Store) (3,000,000 IO)
  - H. Item  $\bowtie$  (Store  $\bowtie$  Sales) (2,800,000 IO)
9. (1 point) True or False: If we have an unclustered B+-tree index on Sales.quantity, the System R optimizer may consider the index scan due to interesting order even if a sequential scan is cheaper.
  - A. True
  - B. False

Consider the above histogram for Sales.quantity and the additional histograms below on Item.price and Item.popularity. Assume that all histogram boundaries are left inclusive and right exclusive. (Simplify all your answers, and use 2 significant digits.)

0-100	100-300	300-900	900-2000	2000-4999
50%	25%	15%	7%	3%

Figure 2: Histogram on Item.price.

0-4	4-6	6-8	8-9	9-10
10%	20%	40%	25%	5%

Figure 3: Histogram on Item.popularity.

10. (2 points) What is the selectivity for the predicate:

$\text{Sales.quantity} \geq 225 \text{ AND } \text{Item.popularity} \geq 2 \text{ AND } \text{Item.popularity} < 6$

11. (2 points) What is the selectivity for the predicate:

$(\text{Item.price} \geq 50 \text{ and } \text{Item.price} < 700) \text{ OR } (\text{Item.popularity} \geq 7)$



Consider the same query above. Suppose the optimizer has narrowed the candidate plans to the following two:

Plan 1: (Item  $\bowtie$  Sales)  $\bowtie$  Store) Estimated total IO: 2,500,000

Plan 2: (Sales  $\bowtie$  Store)  $\bowtie$  Item) Estimated total IO: 4,500,000

In this case, the optimizer will choose plan 1 as its estimated IO is less than that of plan 2. In the lecture, we have been making assumptions that the data distribution is uniform and the columns are independent of each other. However, these assumptions rarely hold in real life. Consequently, the actual IO might differ significantly from the estimated IO.

One way to potentially improve the plan selection is to incorporate feedback; after executing the query using plan 1 and observing the actual IO of this plan, we can feed this information back to the optimizer to let it better optimize executions of the same query in the future.

12. (1 point) Suppose we observe that the actual IO is 5,000,000 for plan 1. If we rerun the same query, which plan will the optimizer choose this time?
  - A. Plan 1
  - B. Plan 2
13. (1 point) Suppose we observe that the actual IO is 2,000,000 for plan 1. If we rerun the same query, which plan will the optimizer choose this time?
  - A. Plan 1
  - B. Plan 2
14. (1 point) True or False: With this feedback mechanism, the optimizer will eventually find the optimal execution plan (in terms of the actual IO, not estimated IO) after running the same query over and over again.
  - A. True
  - B. False

## 4 Transactions and Concurrency Control (12 points)

1. (1.5 points) Fill in the corresponding box on the answer sheet if True.
  - A. If only the last two operations out of many in a committed transaction do not take effect, it is a violation of the Atomicity principle.
  - B. Interleaving transactions is always faster than doing them one at a time.
  - C. All conflict serializable are also serial schedules.
  - D. Transaction A reads from relation W, and Transaction B reads from relation W concurrently. They may possibly read any or all of W. This is considered a conflict.
  - E. Transaction A writes to relation W, and Transaction B writes to relation W concurrently. They may possibly write to any or all of W. This is considered a conflict.

Use the schedule below for the next two questions. Assume both A and B have a value of 100 before T1 begins.

T1	R(B)	R(A)	A=A+100	B=B+A	W(A)	B=B+100	W(B)	A=A+B	W(A)	COM
----	------	------	---------	-------	------	---------	------	-------	------	-----

2. (1 point) Suppose T1 successfully commits.
  - (a) What is the final value of A?
  - (b) What is the final value of B?
3. (2 points) Suppose T1 starts and its execution is **not** atomic. (You may mark none or more than one for the choices below.)
  - (a) Which of the following are possible final states of A?
    - A. 100
    - B. 200
    - C. 600
  - (b) Which of the following are possible final states of B?
    - A. 100
    - B. 300
    - C. 400
4. (4.5 points) Use the schedule below for this question. (You may mark none or more than one for the choices below.)

T1	R(B)						R(C)			R(A)	
T2		R(B)			R(D)						W(A)
T3			W(B)			R(C)					
T4				W(B)				W(C)	W(D)		

- (a) What transactions is T1 pointing to in the conflict graph for the schedule above?

- A. T2
  - B. T3
  - C. T4
- (b) What transactions is T2 pointing to in the dependency graph for the schedule above?
- A. T1
  - B. T3
  - C. T4
- (c) What transactions is T3 pointing to in the dependency graph for the schedule above?
- A. T1
  - B. T2
  - C. T4
- (d) What transactions is T4 pointing to in the dependency graph for the schedule above?
- A. T1
  - B. T2
  - C. T3
- (e) True or False: The schedule above is conflict serializable.
- (f) Which of the following schedules below are conflict equivalent to the schedule above?
- A. T2, T1, T4, T3
  - B. T2, T1, T3, T4
  - C. T1, T2, T3, T4
  - D. T1, T2, T4, T3
5. (3 points) Consider the following schedule, with  $A = 90$  and  $B = 90$  at the start. Mark the statements below that are True.

T1	T2
<pre> read(B) read(A) <b>if</b> (B&gt;100) <b>then</b> A=A+B <b>else</b> A=A+10 write(A) </pre>	<pre> read(A) read(B) <b>if</b> (A&gt;100) <b>then</b> B=A+B  <b>else</b> B=B+10 write(B) </pre>

- A. The schedule is serializable.
- B. The schedule is view serializable.
- C. The schedule is **not** conflict serializable