EECS 484 Sample Final Exam

1. This is a closed book exam. But you are allowed to bring hand-written notes on one double-sided 8.5x11 sheet of paper with you.
2. You have 100 minutes to complete this exam. The points on this exam total to 100. Points roughly indicate the number of minutes it can take to answer a question.
3. There are no intentional typos or syntax errors in the exam. If you see any, ask us to correct them, if the correction is not obvious. Questions on the exam are about semantics, not syntax.
4. **Enter your answers on the provided scantron sheets. Make sure you also print and bubble in the name and UMID . Use #2 pencils and follow the additional instructions on the next page.**
5. You can use an approved calculator of one of the models that was posted on piazza. Use of any other calculator will be considered a violation of the Engineering Honor Code. No other electronic tools are allowed. You may find it faster to *not* use a calculator on this particular exam. The exam is doable, in instructor's opinion, without a calculator.
6. You can use the exam book for your rough work – that is not graded. Only the scantron sheets are going to be graded. There is a blank page at the end.
7. Please sign the honor pledge, turn in this exam and scantron sheets, and show your picture ID. Thank you.

Your exam room number: _____

Person to your left (if none, left-most person behind you): _____

Person to your right (if none, right-most person ahead of you): _____

Make and model of your calculator (if none, write None): _____

Write a brief explanation if you are not filling in a name (e.g., "Left-most person in the back row"). Not filling out the above will lead to a deduction of 2 points.

Honor Code Pledge: I have not received or given aid on this exam and have not concealed any violation of the Honor Code
Your signature: _____

Your name: _____

Your uniquname: _____

Additional scantron instructions

1. Fill in the bubbles darkly and completely. Also, erase any stray marks or changed answers as well as you can since the scantron sheets are evaluated by a machine.
2. Do not fold, bent, or roll up the scantron sheet.
3. It is important that you only fill **one bubble per question**. Even if you think multiple choices are valid, select only one choice because scantron will otherwise reject your answer. Let us know after the exam if you felt a question had more than one valid answer.
4. Request to manually grade the exam will normally lead to a deduction of 20 points and it will delay the grading till next term. The exam is intended to be auto-graded. Fill in the scantron sheets carefully.

Page	Points
3	4
4	6
5	6
6	6
7	4
8	5
9	7
10	5
11	6
12	4
13	9
14	5
15	6
16	6
17	10
18	11
Total:	100

Relational database security (SQL standard):

1. (2 points) User A owns table T1 and user B owns the table T2. User C wants to run the following query:

   ```
   SELECT * FROM T1, T2 WHERE T1.a = T2.b;
   ```

   Which of the following statements will A and B need to be execute to grant the appropriate permissions to C to run the query? Select the best answer.

   A. A does the following: GRANT SELECT ON T1 to B;

   B. B does the following: GRANT SELECT ON T2 to C;

   C. Both of the above

   D. None of the above choices are correct.

2. (2 points) Role A grants a right to insert tuples on a table T to role B. Role B in turn grants that right to role C. Subsequently, role A successfully revokes the right to insert tuples on table T from role B. What happens to the right of role C to insert tuples in the table T?

   A. The right of role C to insert tuples in T is definitely revoked.

   B. The right of role C to insert tuples in T is revoked only if role C wasn't also granted the right by a role other than B.

   C. The right of role C to insert tuples in T is not revoked.

Database semantics:

3. (2 points) Which two of the ACID properties is ARIES protocol most designed to support?

    A. Durability and isolation

    B. Isolation and consistency

    C. Atomicity and isolation

    D. Atomicity and durability

4. (2 points) Suppose that a table T exists in Postgres – a traditional database with two tuple t1 and t2. Two SQL commands are issued simultaneously, the first of which updates both t1 and t2 and the second of which does the following query:

    ```
    SELECT * FROM T;
    ```

    What can one say about the result from the second query, under standard Postgres semantics? Select the best answer:

    A. It will print out old t1 and old t2

    B. It will print out one old tuple and one new tuple

    C. It will print new t1 and new t2

    D. All of the above are possible

    E. Only A. and C. are possible

5. (2 points) Now consider the equivalent of the above commands in mongoDB (with appropriate mapping to collections, documents, and commands in mongoDB). What can one say about the result from the second operation under standard mongoDB semantics? Select the best answer:

    A. It will print out old t1 and old t2

    B. It will print out one old tuple and one new tuple

    C. It will print new t1 and new t2

    D. All of the above are possible

    E. Only A. and C. are possible

Transactions and locking:

Time	T1	T2	T3
0		W(X)	
1			R(X)
2			Commit
3	W(Y)		
4	Commit		
5		R(Y)	
6		W(Z)	
7		Commit	

Considering the above schedule, answer the following questions:

6. (2 points) Is the schedule serializable?
   A. True
   B. False
7. (2 points) Is the schedule conflict-serializable?
   A. True
   B. False
8. (2 points) Is the schedule recoverable?
   A. True
   B. False

2PL Locking and schedules:

9. (1 point) A schedule implemented by following a non-strict 2PL locking protocol is guaranteed to avoid cascading aborts?

    A. True

    B. False

10. (1 point) A schedule implemented by following a non-strict 2PL locking protocol is guaranteed to be recoverable?

    A. True

    B. False

11. (4 points) Consider the following complete schedule of transactions T1, T2, T3.

Time	T1	T2	T3
0	R(X)		
1		R(X)	
2			W(Y)
3	Commit		
4			W(Z)
5			R(X)
6		W(X)	
7			Commit
8		W(Z)	
9		Commit	

Consider that the system implements a strict 2PL locking. Is it possible for the above schedule to arise? If not, indicate the first discrepency.

A. The schedule can arise

B. No, there is a discrepancy in time slot 1

C. No, there is a discrepancy in time slot 5

D. No, there is a discrepancy in time slot 6

E. No, there is a discrepancy in time slot 8

Generating Transaction Schedule: Suppose the following transactions all arrive at time 0, in the order T1, T2, T3 and we use strict 2PL for scheduling the transactions:

```
T1: W(X), R(X), commit
T2: R(X), W(X), commit
T3: R(X), W(X), commit
```

The operations within the transactions are done in round-robin fashion under similar rules as on Homework 5 problems, which are repeated for your reference:

1. Runnable transactions run in round-robin fashion. In other words, each transaction does one step (either R or W operation or COMMIT), goes to the end of the runnable queue, and then cedes to the next runnable transaction.
2. If a transaction blocks on a lock request, it goes to the end of the corresponding lock queue. This takes 0 time and is not considered a step in the transaction.
3. COMMIT is a separate operation and releases all locks.
4. Assume that locks are released in the order they were acquired and they are released immediately on COMMIT and transferred to the first waiting transaction. If a write lock is released and a reader is the first one in the queue, all waiting readers are granted the lock in the order in which they were waiting, irrespective of their positions in the queue. They become runnable and put at the end of the runnable queue as they are woken up. All this takes 0 time.

12. (2 points) In what time slot does T1 commit? **Note that the earliest time slot is 0.** If it does not commit (e.g., because of a deadlock), select None of the above.

    A. 0 or 1

    B. 2

    C. 3

    D. 4

    E. None of the above

13. (2 points) In what time slot does T2 commit? If it does not commit (e.g., because of a deadlock), select None of the above.

    A. slots 0, 1 or 2

    B. slots 3 or 4

    C. slots 5 or 6

    D. slots 7 or 8

    E. None of the above

Database Recovery

14. (3 points) You are building a new database system. The buffer manager is designed to delay (as much as possible) all writes of pages to disk due an abundance of memory. Which recovery phases will likely take the longest to execute (a.k.a see the most degradation in performance?)

    A. Analysis

    B. Redo

    C. Undo

    D. The recovery will take equal time or less time.

ARIES Protocol:

Consider the following recovery log. Assume that before step 1 in the LSN, the Transaction Table and Dirty Page Table are empty. Assume there none of the dirty pages are forced out to disk during this log.

LSN LOG

```
1. start T1
2. start T2
3. update: T1 writes to P1
4. update: T2 writes to P2
5. start T3
6. update: T3 writes to P2
7. update: T1 writes to P1
8. update T2 writes to P2
9. T1 commit
10. T1 end
11. begin checkpoint
12. end checkpoint
13. update: T3 writes to P1
14. update: T2 writes to P3
15. update: T3 writes to P2
16. CRASH
```

15. (2 points) At which LSN will the ANALYSIS phase start processing the log?

    A. LSN 1

    B. LSN 3

    C. LSN 11

    D. LSN 16

    E. None of the above

16. (3 points) Which of the following (page, LSN) pairs will be in the Dirty Page Table that is restored at the start of the Analysis Phase of recovery?

    A. (P1,7), (P2,8)

    B. (P1, 3), (P2,4)

    C. (P1, 13), (P2, 15), (P3,14)

    D. (P1, 3), (P2,4), (P3, 14)

    E. None of the above

17. (2 points) The LSN value from which the REDO phase will start processing the log is (*all ranges inclusive*):

    A. between 11 and 13

    B. between 1 and 3

    C. between 5 and 10

    D. between 14 and 16

    E. None of the above

18. (2 points) If we always had a no-steal policy, what changes could be made to the ARIES protocol:

    A. Analysis phase will not be required

    B. Redo phase will not be required

    C. Undo phase will not be required

    D. None of the above. All the three phases will still be required

Sort-Merge Joins:

The following three questions are related. Assume you have a relation R1 that is **sorted** on attribute A and a relation R2 that is **sorted** on attribute B. Also assume that relation R1 consists of n pages and relation R2 consists of m pages. The optimizer decides to do a sort-merge-join when joining R1 and R2 on R1.A = R2.B, and it takes advantage of already sorted relations to the extent possible, avoiding any steps that can be eliminated from the algorithm to reduce the I/O. Ignore the cost of writing the final result in the following problems.

19. (2 points) What is the best-case performance of doing the join for the above situation?

    A. $O(n + m)$

    B. $O(n * m)$

    C. $O(nlog(n) + mlog(m))$

    D. None of the above.

20. (2 points) What is the worst-case performance of the join for the above situation?

    A. $O(n + m)$

    B. $O(n * m)$

    C. $O(nlog(n) + mlog(m))$

    D. None of the above.

21. (1 point) What characteristic of the input relations would cause the worst-case performance?

    A. R1.A values and R2.B values are all different from each other.

    B. R1.A values are all identical and they match exactly one row of R2 on attribute B.

    C. R1.A values are all equal to each other, R2.B values are all equal to each other, but R1.A values are different from R2.B values.

    D. R1.A and R2.B values are all equal to each other

    E. None of the above choices are correct

Index Matching

22. (2 points) Given a B+-tree index on table *T(a, b, c, d)* with the composite search key $< a, b, c >$ and the following SQL query, would the index match the following query? In that case, select Yes as the answer. Only if not, select the best index that matches among the remaining choices for the answer.

```
SELECT * FROM T WHERE a = 10 AND c < 20;
```

    A. Yes, this index matches.

    B. No the index does not match. Instead use a hash-index index on $< a, c >$

    C. No the index does not match. Instead use a B+-tree index on $< a, c >$

    D. No, the index does not match and none of the above choices are correct.

23. (2 points) Given the following SQL query, which index matches the query?

```
SELECT a, b FROM t WHERE (a > 80 AND b < 2) OR (c = 3);
```

    A. B+-tree index on $< a, b >$

    B. Hash index on $< a, b >$

    C. Hash index on $< c >$

    D. B+-tree index on $< a, b, c >$

    E. None of the above choices are correct.

24. (2 points) A relation *R(a,b,c)* has an hash index on the composite search key $< a, b >$. For which of the following queries can one do an index-only scan using this hash index? Select the best answer.

    A. `SELECT a, b FROM R where R.a > 23;`

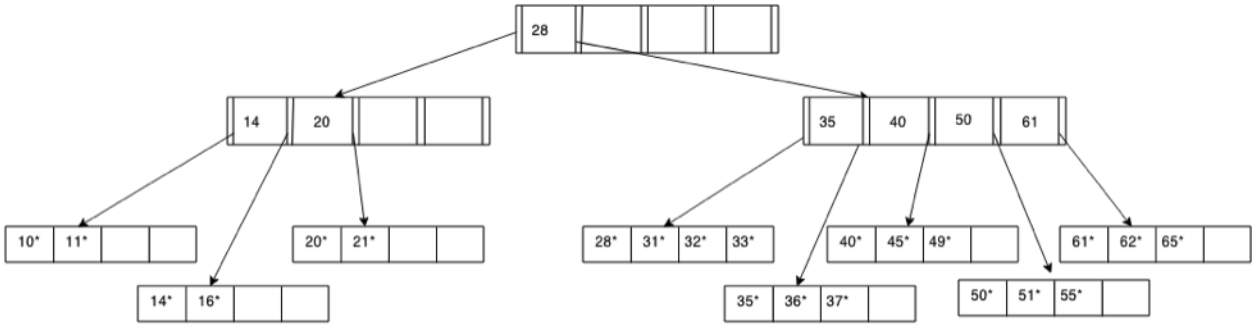
    B. `SELECT b FROM R where R.a = 23;`

    C. `SELECT a FROM R where R.b = 40;`

    D. More than one of the above choices is correct.

    E. None of the above choices are correct.

B+-tree structure and algorithms Consider the following B+-tree providing an unclustered index for a relation *R(id, name)* on a search key *id*, assuming the convention that the left pointer points to values that are strictly less than the key value. Each node in this B+-tree can hold at most four data entries and the leaf level stores entries using **Alternative 2**. Assume that leaf nodes are organized as a **doubly-linked list as usual in a B+-tree** (those pointers are not shown in the diagram for brevity, but they **do exist** and you should take advantage of them as needed.) Also, note that data entries within the leaves are maintained in sorted order.



The following questions are independent of each other, though all based on the above tree. There are no other index structures available on R for all the following problems.

25. (2 points) The maximum fan-out of above tree is 5. Suppose you are given a lare dataset (not the one shown). Compared to the above tree with max fan-out of 5, if instead a tree with a fanout of 100 was used, a search query that utilizes the index will usually require:

    A. Increased page I/O overall

    B. Reduced page I/O overall

    C. About the same page I/O overall

26. (2 points) Assume that the each tuple in relation R occupies 100 bytes. How many bytes are required for all the data records of relation R (ignore unused space in data pages – just count the bytes for the data records.)

    A. 1000 bytes or less

    B. 1001 - 2000 bytes

    C. 2001 - 3000 bytes

    D. 3001 - 4000 bytes

    E. more than 4000 bytes

Understanding the use of index in answering queries:

The following three questions use the same B+-tree diagram as on the previous page.

- Assume that each node in the above B+-tree takes up one page as usual
- Assume unclustered index and each data record of R is on a different page
- Assume that only the root node of the B+-tree is in memory initially and none of the data pages or R are in memory.
- Assume that a page once read can be kept in memory for the duration of the query – so a given page has to be read at most once during a query. Each question is independent however and starts out with only the root of the B+-tree in memory.

For the query given in each of the following three problems, state how many page I/Os must be required to answer the query using the most efficient strategy possible. You can consider strategies that do not use the index. No other index is available.

27. (3 points) `SELECT name FROM R where id >= 15 and id < 42;`

    A. Less than 5 page I/Os

    B. Between 5 and 6 (inclusive) page I/Os

    C. Between 7 and 8 (inclusive) page I/Os

    D. Between 9 and 15 (inclusive) page I/Os

    E. More than 15 page I/Os

28. (3 points) `SELECT id FROM R where id >= 15 and id < 42;`

    A. Less than 5 page I/Os

    B. Between 5 and 6 (inclusive) page I/Os

    C. Between 7 and 8 (inclusive) page I/Os

    D. Between 9 and 15 (inclusive) page I/Os

    E. More than 15 page I/Os

29. (3 points) `SELECT id FROM R where (id == 40) OR (name = Joe);`

    A. Less than 5 page I/Os

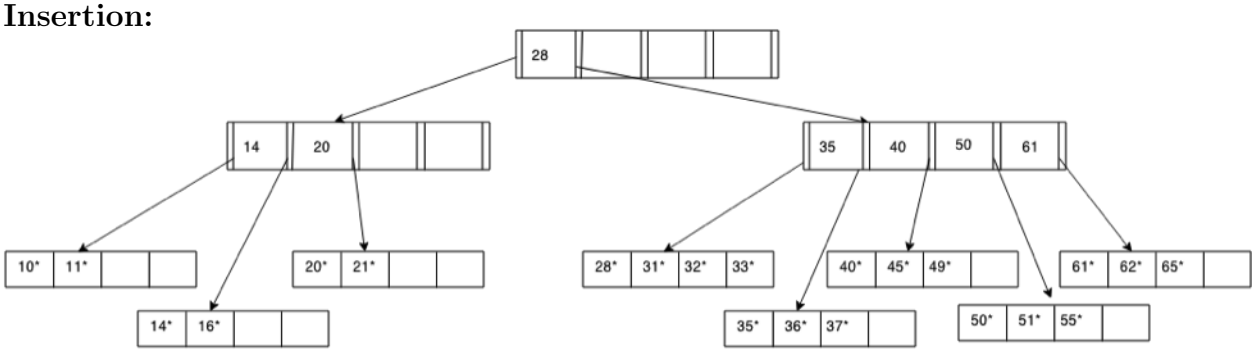
    B. Between 6 and 10 (inclusive) page I/Os

    C. Between 11 and 15 (inclusive) page I/Os

    D. Between 16 and 20 (inclusive) page I/Os

    E. More than 20 page I/Os

Insertion:



Consider the B+-tree displayed above. Assume that the insert algorithm **does not** consider redistributing entries. Further, assume that when a node is split, the resulting **left** node will have equal or more values than the resulting right node (e.g. a 3,2 split rather than than a 2,3 split). Finally, assume that the value that is copied up from the split of a leaf node is the smallest of the values from the right child. For a split of a non-leaf node, middle value is pushed up.

Consider the B+-tree after inserting a data entry with the key value 30. Answer the following questions about the resulting tree.

30. (3 points) The total number of nodes (including the root and leaves) in the resulting B+-tree is

    A. 11

    B. 12

    C. 13

    D. 14

    E. 15

31. (2 points) The minimum of the number of children that exist for a non-leaf node in the resulting tree will be:

    A. 2

    B. 3

    C. 4

32. (3 points) Each node in the tree (whether it is a leaf or a non-leaf node) corresponds to a page. Upon an insert, these index nodes must be updated and, upon splits, new pages added and written to. What is the minimum number of pages in the B+-tree that will need to be written or updated as a result of the insert of 30 in the original tree? Do not count the write to any data pages, since they are outside the B+-tree. Do count the writes to both leaf and non-leaf pages. Note: The minimum possible answer is 1 to any such question since at least the leaf node must be updated in any insert.

    A. 3 or less

    B. 4

    C. 5

    D. 6

33. (3 points) If instead of not allowing redistribution, redistribution were allowed upon insert, how many nodes (both leaf and non-leaf) would the resulting tree have had upon inserting 30 in the original tree?

    A. 11

    B. 12

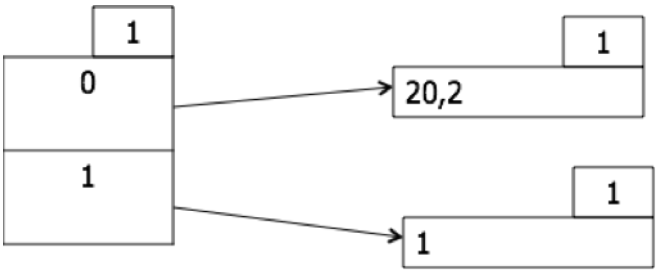
    C. 13

    D. 14

    E. 15

Extendible Hashing:

Consider the following instance of an extendible hash index. The directory is shown on the left, and the pages containing data entries are shown in the right. Use the following hash function: $hashvalue = (key \bmod 2^d)$, where d is the global depth of the index. Assume that for this index the bucket capacity is 2 entries. Note that the bucket at 0 already has two data entries corresponding to 20 and 2.



Insert the entries **32, 16, and 5** into the above index. Answer the following questions on the resulting structure:

34. (2 points) What is the global depth in your index after the insertions?

    A. 1

    B. 2

    C. 3

    D. 4

    E. 5

35. (2 points) What is the minimum local depth in your index after the insertions?

    A. 1

    B. 2

    C. 3

    D. 4

    E. 5

36. (2 points) Which of the following data entries is in the same bucket as data entry 5?

    A. 23

    B. 1

    C. 20

    D. 16

    E. None of the above, or 5 will be alone in its own bucket

37. (2 points) What are the number of pages with entries in your index, excluding the directory?

    A. 3

    B. 4

    C. 5

    D. 6

    E. 7

38. (3 points) If 23 were added next (after adding 32,16, and 5), which data entry will it end up with?

    A. It will be in a bucket by itself

    B. With 1 only

    C. With 5 only

    D. With 1 and 5

    E. None of the above.

39. (5 points) *(Broader question not specific to this dataset):* Suppose that you already have a hash index using extendible hashing on relation R on attribute a and another hash index using extendible hashing on relation S on attribute b. Both used exactly the same hash function and both ended up with the same global depth. Both use **Alternative 1**, i.e., the k* data entry values in each bucket is simply the entire record (tuple). A bucket consists of one page and, luckily, all the buckets ended up being full. What is the minimum number of I/O operations in pages required to join the two relations on $R.a = R.b$ using the best possible strategy one can design, taking advantage of the two available hash indexes, if it helps? Assume that only **three** buffer pages are available to do the join. Don't count the I/O for writing the final result. Assume that $|R|$ and $|S|$ (number of pages for R and S, respectively) are large quantities.

    A. Around $|R| + |S|$

    B. Around 2*($|R| + |S|$)

    C. Around 3*($|R| + |S|$)

    D. It will be much higher than above choices (e.g., big-Oh will be higher or a larger multiplicative factor will be required).

Query Planning

40. (5 points) You are given two relations X and Y. X has 20,000 tuples and 200 pages. Y has 10,000 tuples and 100 pages. Your database has 10 buffer pages available. How many READ I/Os (in terms of **pages**) must be done during Grace Hash Join? Assume that the partitioning is even across buckets, and if multiple rounds of partitioning are required, same number of rounds of partitioning is used for both relations. Pick the **closest** answer.

    A. 300 pages will be read.

    B. 600 pages will be read.

    C. 900 pages will be read.

    D. 1200 pages will be read.

    E. 10,000 pages will be read.

41. (3 points) Consider the following join methods on the tables in the previous problem

    1. Simple nested loops join; 2. Block nested loops join; 3. Sort-merge join

    Which of the above methods will produce the fewest number of WRITE I/O operations in terms of pages (i.e., ignore read I/O operations)? Select the best answer.

    A. 1. only

    B. 2. only

    C. 3. only

    D. 1 and 2

    E. 2 and 3

42. (3 points) You are given two relations X and Y. X uses 200,000 pages and Y uses 10 pages (both with 10 tuples per page). Your database has 100 buffer pages available. Which join algorithm (among the following three methods) should be picked by a query planner to minimize the total number of I/O operations in terms of pages (counting both reads and writes)?

    A. Simple-nested loop.

    B. Block-nested loop.

    C. Sort-merge join

Page intentionally left blank

Page intentionally left blank