This exam is open book and notes. You have 180 minutes to complete it. There are 8 questions. The exam is worth 180 points total.

Leland login ID:  
Name:  

The Honor Code is an undertaking of the students, individually and collectively:

1. that they will not give or receive aid in examinations; that they will not give or receive unpermitted aid in class work, in the preparation of reports, or in any other work that is to be used by the instructor as the basis of grading;
2. that they will do their share and take an active part in seeing to it that others as well as themselves uphold the spirit and letter of the Honor Code.

The faculty on its part manifests its condense in the honor of its students by refraining from proctoring examinations and from taking unusual and unreasonable precautions to prevent the forms of dishonesty mentioned above. The faculty will also avoid, as far as practicable, academic procedures that create temptations to violate the Honor Code. While the faculty alone has the right and obligation to set academic requirements, the students and faculty will work together to establish optimal conditions for honorable academic work.

I acknowledge and accept the Honor Code.

Signed:  

<table>
<thead>
<tr>
<th>Problem</th>
<th>Points</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>30</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>36</td>
</tr>
<tr>
<td><strong>Total:</strong></td>
<td></td>
<td><strong>180</strong></td>
</tr>
</tbody>
</table>
Problem 1 (10 points)

State whether the following statements are **TRUE** or **FALSE**.

- **False**  a) There exists a pair of transactions such that regardless of the order of execution of their actions, a deadlock must occur at some point in the execution.

- **True**  b) For any predicate $p$: $\sigma_p(R \cup S) \equiv \sigma_p(R) \cup \sigma_p(S)$.

- **False**  c) For any set of attributes $x$ and any predicate $p$: $\pi_x[\sigma_p(R)] \equiv \sigma_p[\pi_x(R)]$.

- **False**  d) Using a B+Tree index to retrieve records in sorted order is always faster than performing a two-pass external merge-sort.

- **False**  e) In REDO logging, all actions of a transaction must be flushed to disk before the commit record is written to the log.

- **True**  f) In UNDO/REDO logging, no modified data can be written to disk before the corresponding log records have been written to disk.

- **False**  g) The RAID disk redundancy techniques protect against catastrophic failure.

- **False**  h) Strict two-phase locking is the only concurrency control technique that guarantees to produce conflict serializable schedules that are recoverable and avoid cascading rollback.

- **False**  i) All strict schedules are serial.

- **True**  j) If each dimension in a 4-dimensional data cube has 3 possible values, then the extended data cube will contain up to 256 data entries, including both the base data entries and the aggregate entries.
Problem 2  (12 points)

Consider a linear hashing index with buckets that can hold up to X records (or key/ptr. pairs). Say that \( m \), the id of the last active bucket, is increased whenever the “utilization” exceeds 80%. In other words, if N is the total number of records stored in the index, then \( m \) is increased whenever \( N/(X*(m+1)) > 0.8 \).

a) Illustrate the final state of a linear hashing index with \( X = 2 \) when records with the following hash values are inserted, in order:
\[ 0011, 1100, 0101, 1011, 0010, 1101, 0111, 1001, 0001 \]
Assume that the index starts out empty with one bucket allocated and with \( m = 0 \).

\[
\begin{array}{c|c|c|c}
000 & & & \\
001 & 1001 & \text{Overflow} & \\
010 & 0010 & & \\
011 & 0011 & 1011 & \text{Overflow} \\
100 & 1100 & & \\
101 & 0101 & & 1101 \\
\end{array}
\]

b) In general, for an index with \( N \) records, what is the minimum number of active buckets possible? Give your answers in terms of \( N \) and \( X \).

\[ N / (X \times 0.8) \]

c) In general, for an index with \( N \) records, what is the maximum number of active buckets possible? Give your answers in terms of \( N \) and \( X \).

\[ N / (X \times 0.8) + (N / X) - 1 \]

Main Buckets \( = N / (X \times 0.8) \)
Over flow Buckets \( = (N / X) - 1 \)

Consider the case in which all the \( N \) keys are hashed to the same value.
Problem 3  (10 points)

Consider the following list of variable-length string keys (given in alphabetical order):

Panda, Panther, Penguin, PolarBear, Puma, PurplePeopleEater

Consider building a B+Tree with these keys. Say that the nodes of the B+Tree can only hold a total of 20 characters, plus all necessary pointers and header information. This upper bound on node capacity applies to both leaf and non-leaf nodes. There is no lower limit on node capacity (i.e., no minimum fullness).

a) Draw a B+Tree that contains the keys given above and has height at most 3 (including the leaf level).

b) Using the key compression technique just described, draw a B+Tree that contains the keys given above and has height at most 2 (including the leaf level). Make sure that no leaf or non-leaf node contains more than 20 characters total.
Problem 4 (12 points)

Consider the following single transaction running in isolation:

\[ T = r(A) r(B) w(A) w(B) r(C) w(C) w(D) \]

Say that the system crashes and when it restarts we have an opportunity to examine the log. A number of scenarios are given below. In each scenario, the logging method and log are given. For each scenario, determine which writes to data elements \textbf{must} be reflected in the database on disk, and which writes \textbf{must not} be reflected on disk. Indicate that a write to data element X is in one of these categories by writing “X” in the appropriate box, along with any other data elements that fit that category. If no data elements fit a category, simply write “empty” in the box. (Blank boxes get no credit.)

a) UNDO logging: \[ \text{log} = \text{<Start T>} <T, A, 5> <T, B, 7> \]

\begin{tabular}{ccc}
\textbf{must:} & \textbf{empty} & \textbf{must not:} C, D \\
\end{tabular}

b) UNDO logging: 
\[ \text{log} = \text{<Start T>} <T, A, 5> <T, B, 7> <T, C, 2> <T, D, 9> <\text{Commit T}> \]

\begin{tabular}{ccc}
\textbf{must:} & A,B,C,D & \textbf{must not:} empty \\
\end{tabular}

c) REDO logging: 
\[ \text{log} = \text{<Start T>} <T, A, 1> <T, B, 2> <T, C, 5> \]

\begin{tabular}{ccc}
\textbf{must:} & empty & \textbf{must not:} A,B,C,D \\
\end{tabular}

d) REDO logging: 
\[ \text{log} = \text{<Start T>} <T, A, 1> <T, B, 2> <T, C, 5> <T, D, 0> <\text{Commit T}> \]

\begin{tabular}{ccc}
\textbf{must:} & empty & \textbf{must not:} empty \\
\end{tabular}

e) UNDO/REDO logging with nonquiescent (fuzzy) checkpointing: 
\[ \text{log} = \text{<Start T>} <T, A, 5, 1> <\text{Begin Checkpoint (T)}> <T, B, 7, 2> \]

\begin{tabular}{ccc}
\textbf{must:} & empty & \textbf{must not:} C, D \\
\end{tabular}

f) UNDO/REDO logging with nonquiescent (fuzzy) checkpointing: 
\[ \text{log} = \text{<Start T>} <T, A, 5, 1> <\text{Begin Checkpoint (T)}> <T, B, 7, 2> <\text{End Checkpoint}> <T, C, 2, 5> <T, D, 9, 0> <\text{Commit T}> \]

\begin{tabular}{ccc}
\textbf{must:} & A & \textbf{must not:} empty \\
\end{tabular}
Problem 5 (30 points)

Consider the following schedule involving three transactions T1, T2 and T3:

<table>
<thead>
<tr>
<th></th>
<th>T1</th>
<th>T2</th>
<th>T3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>W(A)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>R(B)</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>R(C)</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
<td>W(B)</td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>W(B)</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
<td>W(C)</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>R(C)</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>W(B)</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>W(C)</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td></td>
<td>R(A)</td>
</tr>
</tbody>
</table>

a) Draw the precedence graph for this schedule.

```
T1  T3
  T2
```

b) Is this schedule conflict serializable? Why or why not? If it is conflict serializable, give the equivalent serial schedule (just write the order of the transactions).

No because Precedence Graph has cycle.

c) Write down all instances where one transaction “reads from” another transaction.
(If T2 reads from T1, write T1 → T2.)

T1 → T3 (A)

T3 → T2 (C)

d) Is this schedule view serializable? Why or why not? Give the equivalent serial schedule if one exists (just write the order of the transactions).

Yes, W1(B) & W3(B) has no effect on the transactions and neither on final state of database

T1 → T3 → T2
Now swap lines 4 and 5 in the schedule.

e) Is the new schedule conflict serializable? Why or why not? If it is conflict serializable, give the equivalent serial schedule (just write the order of the transactions).

Yes. Now Precedence Graph is acyclic.

T1 → T3 → T2

f) Is the new schedule view serializable? Why or why not? Give the equivalent serial schedule if one exists (just write the order of the transactions).

Yes. Any Conflict Serializable schedule is View Serializable. The serial schedule is

T1 → T3 → T2
Problem 6 (30 points)

Consider the following two transactions:

T1 = r1(A) w1(A) r1(B) w1(B)
T2 = r2(B) r2(A) w2(A) w2(B)

Say that exclusive lock and unlock actions are inserted by the scheduler, resulting in the following annotated transactions:

T1 = lx1(A) r1(A) w1(A) lx1(B) r1(B) w1(B), AFTER COMMIT: u1(A) u1(B)
T2 = lx2(B) r2(B) lx2(A) r2(A) w2(A) w2(B), AFTER COMMIT: u2(A) u2(B)

Say these two transactions are executed concurrently.

a) Is conflict serializability guaranteed? Why or why not?

Yes it is guaranteed because schedules are 2 PL.

b) Is deadlock possible? If so, then assuming that T1 starts first, which transaction(s) would be rolled back (aborted) under the wait-die deadlock prevention scheme?

Yes Deadlock is possible and T2 would roll back.

c) Is cascading rollback possible? If not, explain why not. If so, show a scenario that results in cascading rollback.

Not possible since it Transactions follow Strict 2 PL.
Now, say that lock and unlock actions are inserted in the following way instead:

\[ T1 = lx_1(A) \ r_1(A) \ w_1(A) \ lx_1(B) \ u_1(A) \ r_1(B) \ w_1(B) \ u_1(B) \]
\[ T2 = lx_2(A) \ lx_2(B) \ r_2(B) \ r_2(A) \ w_2(A) \ u_2(A) \ w_2(B) \ u_2(B) \]

\[ \text{d) Is conflict serializability guaranteed? Why or why not?} \]

\[ \text{Yes it is guaranteed because schedules are 2 PL.} \]

\[ \text{e) Is deadlock possible? If so, then assuming that } T1 \text{ starts first, which transaction(s) would be rolled back (aborted) under the } \text{wound-wait} \text{ deadlock prevention scheme?} \]

\[ \text{Not possible because the locks are acquired in an ordered manner.} \]

\[ \text{f) Is cascading rollback possible? If not, explain why not. If so, show a scenario that results in cascading rollback.} \]

\[ \text{Yes. Consider scenario in which} \]
\[ lx_2(A) \ lx_2(B) \ r_2(B) \ r_2(A) \ w_2(A) \ u_2(A) \ lx_1(A) \ r_1(A) \ w_1(A) \ w_2(B) \ u_2(B) \]

\[ \text{Here } T_1 \text{ is reading element, } A \text{ which has been written by } T_2, \text{ which is not committed} \]
Problem 7 (40 points)

This problem examines query processing algorithms for aggregation with grouping. Say relation R has two columns A and B, and the following query needs to be answered:

```
SELECT    A, HIST(B)
FROM       R
GROUP BY   A
```

where HIST() is the aggregation function to compute a histogram. Suppose that relation R has size B(R) blocks, and that M blocks of memory are available. Furthermore, suppose there are N unique values for column A (i.e., V(R, A) = N) and therefore N groups, so the answer will consist of N histograms. For simplicity, assume that storing one histogram requires one block of memory. Furthermore, assume that the number of groups (N) is known in advance. Consider three alternative algorithms for performing the grouping operation, called RAW, SORT, and HASH.

RAW algorithm: Initialize N histograms in memory. Read R from the disk, and for each tuple of R, update the appropriate histogram. At the end, write out the histograms.

SORT algorithm: Phase I (identical to Phase I of merge sort): Read in memory-sized chunks of R, sort them on A using quicksort, and write out the sorted runs. Phase II: Read in and merge the sorted runs. Reserve one block of memory for a histogram. Every time a tuple exits the merge process, update the histogram. Whenever a tuple with a new R.A value comes in, write out the current histogram and initialize a new histogram. Since the merged tuples are ordered on R.A, the groups will be contiguous and the histograms can be created one at a time.

HASH algorithm: Phase I (identical to Phase I of hash join for one relation): Hash R into buckets by applying a hash function to column A and write out the buckets to disk. Phase II: Read one bucket at a time – for each bucket, construct the histograms for the groups found in that bucket and then write out the histograms.

(a) How much memory is required to execute the RAW algorithm? Ignore the extra block of memory required for an input buffer.

\[ N \]

(b) What is the total cost of the RAW algorithm (including the cost of writing out the histograms)?

\[ B(R) + N \]
c) How much memory is required to execute the SORT algorithm? Ignore the extra block of memory required to construct the histograms one at a time.

\[
\text{\# runs} = \frac{B(R)}{M}
\]

Need \( M > \text{\# runs} \)

\( M \geq \frac{B(R)}{M} \)

\( M \geq \sqrt{B(R)} \)

d) What is the total cost of the SORT algorithm (including the cost of writing out the histograms)?

\[ 3 \ B(R) + N \]

e) After Phase I of the HASH algorithm, how many groups will there be in each bucket (assume the hash function distributes records evenly)? Ignore the extra block of memory required for an input buffer.

\[ \frac{N}{M} \]

f) Let \( G \) be the number of groups per bucket (i.e., the answer for part (e)). In terms of \( G \), how much memory would be required to execute Phase II of the HASH algorithm? Ignore the extra block of memory required for an input buffer.

\[ G \]

g) How much memory is required to execute the entire HASH algorithm? As before, ignore the extra block of memory required for an input buffer in Phases I and II.

\[ M \geq G \]

\[ M \geq \frac{N}{M} \]

\[ M \geq \sqrt{N} \]

h) What is the total cost of the HASH algorithm (including the cost of writing out the histograms)?

\[ 3 \ B(R) + N \]
Problem 8  (36 points)

Consider a crime lab database of criminals. Say there are two tables: H(picture, favorite_bar, …) has information about human criminals and D(breed, favorite_bar, …) has information about dog criminals. There are 10,000 known human criminals (i.e., \(|H| = T(H) = 10,000\)) and 50,000 known dog criminals (i.e., \(|D| = T(D) = 50,000\)). Assume that 10 dog records fit on a block (so \(B(D) = 5000\)), but only one human record fits on a block because human records contain a picture (so \(B(H) = 10,000\)). Assume also that joined human-dog records take up a whole block. Finally, assume that the crime lab computer has \(M = 102\) blocks of memory available.

A witness saw a man with a mustache rob a liquor store with a dog accomplice. No information about the dog was observed, but its human partner had a mustache. To compile a list of suspects, the crime lab staff decided to run the following query, which finds pairs consisting of mustache-wearing humans and dogs that frequent the same bar:

\[
\text{SELECT} \quad * \\
\text{FROM} \quad H, D \\
\text{WHERE} \quad H.\text{favorite}_\text{bar} = D.\text{favorite}_\text{bar} \text{ AND } \text{HasMustache}(H.\text{picture})
\]

a) Write down two logical query plans for this SQL query. Neither plan should contain a cross-product (Cartesian product), and both plans should have \(H\) as the left (outer) input to the join. The first one (call it \(P\)) should have the selection pushed down. The second one (call it \(Q\)) should have the selection after the join.
Say that there are a lot of bars in town, so for a given human-dog pair, the probability that they have the same favorite bar is $10^{-6}$. Furthermore, say that about 1% of human criminals have mustaches. Assume that whether or not a person has a mustache does not influence his choice of bars (i.e., the two predicates are statistically independent). Finally, assume the HasMustache() predicate requires some expensive image processing with a cost equivalent to $N$ IO’s.

Assume that block (chunk) nested-loops join is the only physical operator implemented in the crime lab database. Recall that in block nested-loops join, the left input is used in the outer loop, and the right input is used in the inner loop. During processing, $M-2$ blocks of the outer relation are read into memory, and then the entire inner relation is streamed through memory to find matches. This processes is repeated once for every chunk of size $M-2$ of the outer relation.

Assume that selection operators are applied “on the fly.” This means that while the relation is being read from disk and placed into memory, the predicate is evaluated and tuples that fail to satisfy the predicate are eliminated immediately, making room for more tuples.

b) Estimate the cost of plans P and Q. Do not include the cost of writing out the final result. Your answer should be in terms of $N$, the cost of the HasMustache() predicate. Indicate clearly which cost formula is for P, and which is for Q.

Cost (Q) = \((B(H) / 100) \times B(D) + B(H) + (M \times D / 1000000) \times N\) 
= \(500,000 + 10,000 + 500 N\) 
= \(510,000 + 500 N\)

Cost (P) = \(((1\% \times B(H)) / 100) \times B(D) + B(H) + |H| \times N\) 
= \(5000 + 10,000 + 10,000 N\) 
= \(15,000 + 10,000 N\)

c) For what values of $N$ is plan P expected to be cheaper than plan Q?

Cost(Q) > Cost (P)

$N < 52.1$

d) Why, for certain values of $N$, is it a good idea to push down the selection? Give the intuition as a qualitative statement. Why, for certain values of $N$, is it not a good idea to push down the selection? Again, give the intuition as a qualitative statement.

If $N$ is small, Perform selection first to make Join cheaper.

If $N$ is big, Perform Join first to make selection cheaper (because of fewer tuples participating in selection)
END OF EXAM – the remaining pages may be used for scratch paper …