CISC437/637 Database Systems Final Exam

You have from 3:30pm to 5:30pm to complete the following questions. No notes, no book, no wireless (or wired) devices are allowed. Use the back of the page if you need more space.

Please read the following:

- Many of the questions use the university database example from the book. The next page of the exam has the schema diagram; you may tear it out to use as a reference.
- Unless otherwise stated, always assume there is a primary B+-tree index on the primary key and secondary indexes on foreign keys.
- If you use any additional assumptions (other than the ones given in the problem statement) in your solution, state them clearly.

Multiple Choice (2 points each; 38 total)

1. Which of the following is *always* true (not just true of MySQL)?
   (a) a primary index is always on the primary key
   (b) secondary indexes are always on foreign keys
   (c) a primary index is always accompanied by a file stored sequentially on the indexed field(s)
   (d) a and b

2. What does BASE stand for in the context of "big data"?
   (a) Basic Analysis and Security Engine
   (b) Basic Availability, Soft state, Eventual consistency
   (c) Building, Antennae, Span, Earth
   (d) Beta-Alumna Solid Electrolyte

3. Which statement best captures the meaning of "consistency" in the context of transactions?
   (a) data updates performed during the transaction persist on disk after it has completed
   (b) transactions are not aware of each other
   (c) every operation in a transaction must succeed in order for the transaction as a whole to succeed
   (d) each transaction must start and finish with a database that is valid for all defined constraints

4. Which statement best captures the meaning of "isolation" in the context of transactions?
   (a) data updates performed during the transaction persist on disk after it has completed
   (b) transactions are not aware of each other
   (c) every operation in a transaction must succeed in order for the transaction as a whole to succeed
   (d) each transaction must start and finish with a database that is valid for all defined constraints

5. Which of the following are good reasons to use a catalog ID (such as CISC437) as a primary key for a table of university courses rather than an auto-incrementing integer ID?
   (a) it ensures that two different courses cannot have the same catalog ID
   (b) it results in course records being stored in alphabetical order by department code, so that all of a department's courses are grouped together on disk
   (c) it allows for faster lookup of course records by the most common way of accessing them
   (d) all of the above

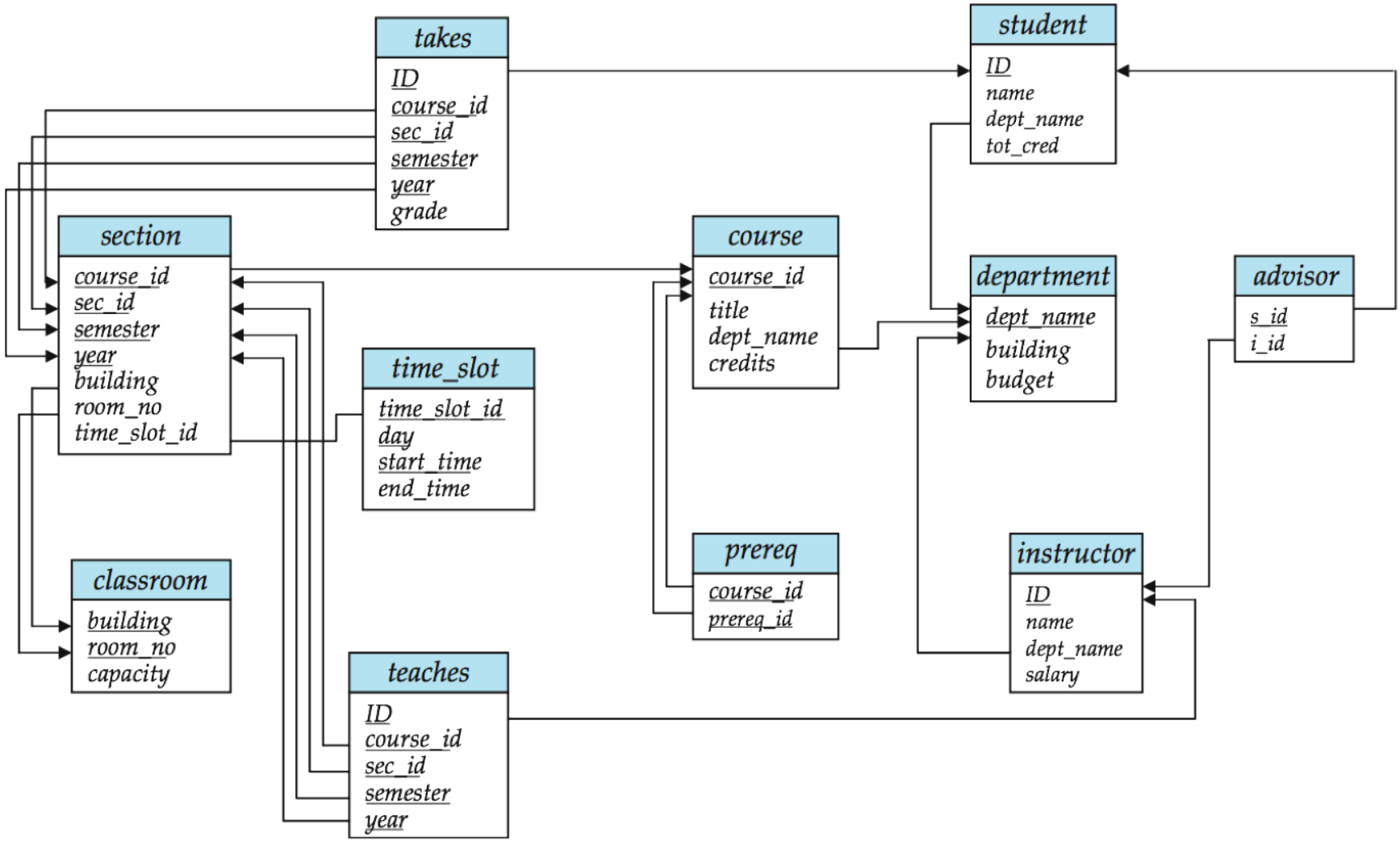

Figure 1: Schema for a university database

6. Suppose a database is read-only—no transactions change any data in the database. Which of the following is true?

   (a) no locking is necessary

   (b) only shared locks are necessary

   (c) only exclusive locks are necessary

   (d) both shared and exclusive locks are necessary

7. If the memory buffer is large enough that uncommitted data changes are never written to disk and can be held in memory indefinitely (that is, stealing is not allowed and forcing is not enabled), does the recovery manager need to support UNDOing transactions? Does it need to suport REDOing transactions?

   (a) UNDO necessary; REDO necessary

   (b) UNDO necessary; REDO not necessary

   (c) UNDO not necessary; REDO necessary

   (d) UNDO not necessary; REDO not necessary

8. What is the purpose of a trigger in a SQL database?

   (a) to keep database logic code with the data it affects rather than in another application

   (b) to define functions that can be called from SELECT statements

   (c) to automatically execute a block of code when certain updates happen in the database

   (d) a and c

9. What is the best definition of a serializable schedule?

   (a) a schedule in which each transaction is allowed to run to completion before another one starts

   (b) a schedule that can be transformed into a serial schedule

   (c) a schedule in which transactions actions are executed in "round robin"

   (d) a schedule with no R-W, W-R, or W-W conflicts

10. What is the main purpose of locking protocols?

    (a) to prevent R-W, W-R, and W-W conflicts when transactions are scheduled in parallel

    (b) to ensure that only one transaction can read a table at a time

    (c) to prevent deadlocks between transactions

    (d) to give the lock manager something to do

11. Why are B+-tree indexes usually preferable to hash indexes?

    (a) B+-tree indexes keep values in sorted order

    (b) B+-tree indexes have predictable storage requirements

    (c) B+-tree indexes allow constant-time access

    (d) a and b

The next 8 questions are based on the following information about file storage and indexes:

```
Student(ID, name, dept_name, tot_cred)
 20,000 records, 100 bytes each
    500 blocks, 40 records per block
     63 blocks for primary index on ID (B+-tree of height 2; 62 leaf nodes)
     53 blocks for secondary index on dept_name (B+-tree of height 2; 52 leaf nodes)
    150 students with dept_name='CISC'

Course(course_id, title, dept_name, credits)
  1,000 records, 200 bytes each
     50 blocks, 20 records per block
      8 blocks for primary index on course_id (B+-tree of height 2; 7 leaf nodes)
      5 blocks for secondary index on dept_name (B+-tree of height 2; 4 leaf nodes)
    600 courses with credits=3

Prereq(course_id, prereq_id)
   800 records, 50 bytes each
    10 blocks, 80 records per block
     1 block for primary index on (course_id,prereq_id) (B+-tree of height 1)

Takes(ID, course_id, sec_id, semester, year, grade)
100,000 records, 40 bytes each
  1,000 blocks, 100 records per block
  1,111 blocks for primary index on (ID,course_id,sec_id,semester,year)
        (B+-tree of height 3; 1,100 leaf nodes)
  1,111 blocks for secondary index on (course_id,sec_id,semester,year)
        (B+-tree of height 3; 1,100 leaf nodes)
```

Assume block size is 4000 bytes and each node in a B+-tree index is stored in one block on disk.

12. What is the best estimate of the minimum number of block reads necessary for the following query?
`SELECT * FROM Student WHERE ID=50505;`
    (a) 1
    (b) $2 + 1$
    (c) $\log_2 500$
    (d) 500

13. What is the best estimate of the minimum number of block reads necessary for the following query?
`SELECT * FROM Student WHERE dept_name='CISC';`
    (a) 500
    (b) $\log_2 500$
    (c) $2 + 150$
    (d) 150

14. Which of the following is likely to be the best approach to evaluating the following query?
`SELECT * FROM Course WHERE credits = 3;`
    (a) full file scan
    (b) index lookup using primary index
    (c) index scan of primary index
    (d) index lookup using secondary index

15. Which of the following is likely to be the best approach to evaluating the following query?

```
SELECT S.name, T.course_id, T.grade FROM Student S JOIN Takes T ON S.ID=T.ID;
```

(a) block nested loop join
(b) indexed nested loop join using primary index on `Student`
(c) indexed nested loop join using primary index on `Takes`
(d) merge join

16. Which *new* secondary index has the best chance of improving processing time for the following query?

```
SELECT S.name, T.course_id, T.grade FROM Student S JOIN Takes T ON S.ID=T.ID AND year=2015;
```

(a) secondary index on `Student.name`
(b) secondary index on `Takes.grade`
(c) secondary index on `Takes.year`
(d) secondary index on `Takes.course_id`

17. What must be done first to support a merge-join between `Takes` and `Course`?

(a) re-sort `Course` on `course_id`
(b) re-sort `Takes` on `course_id`
(c) re-sort `Course` on `ID`
(d) nothing special

18. Which is likely to be the best approach to evaluating the following query?

```
SELECT T.ID, T.course_id, T.grade FROM Takes T JOIN Prereq P
    ON T.course_id = P.prereq_id WHERE P.course_id = 'CISC437';
```

(a) primary index lookup in `Prereq` followed by indexed nested loop join with `Takes` using its primary index
(b) block nested loop join
(c) primary index lookup in `Prereq` followed by indexed nested loop join with `Takes` using its secondary index
(d) primary index lookup in `Takes` followed by indexed nested loop join with `Prereq` using its primary index

19. Which of the following equivalent relational algebra expressions is most likely to give the most efficient evaluation plan for this query?

```
SELECT T.ID, T.course_id, T.grade FROM Takes T JOIN Prereq P
    ON T.course_id = P.prereq_id WHERE P.course_id = 'CISC437';
```

(a) $\pi_{\texttt{T.ID,T.course\_id,T.grade}} \left(\sigma_{\texttt{course\_id=CISC437}} (\texttt{Prereq}) \bowtie_{\texttt{prereq\_id=course\_id}} \texttt{Takes}\right)$
(b) $\pi_{\texttt{T.ID,T.course\_id,T.grade}} \left(\texttt{Takes} \bowtie_{\texttt{prereq\_id=course\_id}} (\sigma_{\texttt{course\_id=CISC437}} (\texttt{Prereq}))\right)$
(c) $\pi_{\texttt{T.ID,T.course\_id,T.grade}} \left(\sigma_{\texttt{course\_id=CISC437}} \left(\texttt{Prereq} \bowtie_{\texttt{prereq\_id=course\_id}} \texttt{Takes}\right)\right)$
(d) $\pi_{\texttt{T.ID,T.course\_id,T.grade}} \left(\sigma_{\texttt{course\_id=CISC437}} \left(\sigma_{\texttt{prereq\_id=course\_id}} (\texttt{Prereq} \times \texttt{Takes})\right)\right)$

Short answer

1. **[15 points]** The following questions ask for estimates of numbers of block reads. You may give arithmetic expressions such as $5 + 5 \times 3$; you do not need to complete the calculations.

   (a) Consider the operation $\sigma_{\texttt{ID=9999}}(\texttt{Student})$. Suppose there are 40,000 student records (with IDs numbered 1 to 40,000) stored over 400 blocks on disk. Estimate the number of block reads necessary to process it if:

      i. There is a primary B+-tree index on `Student.ID` with height 3.

      ii. The `Student` table is stored as a sequential file in order of ID and there is no index.

   (b) Consider the operation $\texttt{Student} \bowtie_{\texttt{ID=s\_id}} \texttt{Advisor}$. There are 40,000 student records stored over 400 blocks on disk, and 40,000 advisor records stored over 100 blocks on disk. Estimate the number of block reads necessary to the query if:

      i. The join is computed using block nested loops.

      ii. The join is computed using indexed nested loops, with a primary B+-tree index on `Advisor.s_id` of height 2.

      iii. The join is computed using merge join.

   (c) **extra credit** Write an expression for the total number of block reads required to process $(\texttt{Instructor} \bowtie \texttt{Advisor}) \bowtie \texttt{Student}$ using sort-merge join for both joins. Assume `Student` is stored in 400 blocks, `Advisor` is stored in 100 blocks, and `Instructor` is stored in 200 blocks. Assume the size of $\texttt{Instructor} \bowtie \texttt{Advisor}$ is 300 blocks.

2. **[15 points]** You have the following query.

```
SELECT * FROM section S JOIN classroom C
           ON S.building = C.building AND S.room_no = C.room_no
        WHERE S.building = 'Kirkbride';
```

Suppose `EXPLAIN`ing this query produces the following output:

id	select type	table	type	possible keys	key	ref	rows
1	SIMPLE	section	ref	PRIMARY, building	building	const	10
1	SIMPLE	classroom	ALL	PRIMARY	NULL		1

(a) Explain in 2 to 3 sentences how the DBMS is processing this query, referencing algorithms we studied in class.

(b) Write the relational algebra expression that matches (as close as possible) MySQL's evaluation plan. Draw the tree diagram for the evaluation plan (you do not have to annotate the tree with algorithms).

(c) Describe one change to the evaluation plan that could be made that might improve processing time for this query.

(d) **extra credit** What information would you need in order to be able to estimate the total cost of the given plan? Make up values and use them to estimate cost.

Transactions [32 points] Below is an *incomplete* schedule for four transactions. The schedule is interrupted by a system crash. Assume every action in the schedule successfully executes before the crash, but not necessarily in the same order they are scheduled.

1. transaction T_A begins
2. R_A(table 1)
3. W_A(table 1)
4. transaction T_B begins
5. R_B(table 1)
6. R_B(table 2)
7. W_B(table 2)
8. R_A(table 2)
9. transaction T_C begins
10. R_C(table 3)
11. W_C(table 3)
12. transaction T_D begins
13. R_D(table 3)
14. R_D(table 4)
15. transaction T_A commit
16. transaction T_C commit
17. system crash

1. The table below represents information about locks that have been granted, released, or that a transaction is waiting on. The locking protocol is rigorous 2PL.

   (a) For each read or write action in the schedule, determine whether the transaction should be granted the appropriate lock. If so, add that transaction number to the list of of transactions holding the lock. Otherwise, add it to the wait list.

   (b) When a lock is released, cross out the transaction that holds that lock in the table. (Assume that granting an exclusive lock automatically releases the shared lock on the same object.)

   (c) When a lock is released, check if there are any transactions on the wait list that should be granted the lock, and if so, give it to the transaction that has waited the longest.

table	shared lock	exclusive lock	waitlist
table 1			
table 2			
table 3			
table 4			

2. Write out scheduled action numbers 1–16 in the order they will actually be executed, starting with $1, 2, 3, 4, ...$

3. The grid below represents the transaction log. Each line is for one log entry. Forcing is disabled and stealing is enabled.

   Directions:

   (a) When a transaction starts, write a transaction-start log entry: `<Ti start>`

   (b) When a transaction updates a table, write a transaction-update log entry. Since I have not provided any actual data values that can be written to the log, write update log entries in the form `<Ti update table n>`.

   (c) When a transaction commits or aborts, write a transaction-end log entry: `<Ti commit>` or `<Ti abort>`.

1. _____	4. _____	7. _____
2. _____	5. _____	8. _____
3. _____	6. _____	9. _____

4. To recover from the crash, the recovery manager will perform a REDO phase starting from the beginning of the log, then an UNDO phase starting from the end. Which log records will need UNDOing? Which will need REDOing?

5. **extra credit** The grid below represents main memory. We can store two full tables in main memory at a time, one in each cell. The memory manager allows stealing and does not force-write changes after commits. Directions:

   (a) For each read action ($R_i(T)$) in the schedule, fill in one of the cells with the name of the table read and the *execution* step number (which may be different from the schedule step number).

      i. If the table is already in memory, it does not have to be read in again, but cross out the previous step number and replace it with the current step number.

      ii. If memory is full, select the least-recently-used table in memory (the one with the lowest associated step number) to swap out and cross it out.

   (b) For each write action ($W_i(T)$) in the schedule, mark the table with a star to indicate it has been updated.
