

Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.830 Database Systems: Fall 2009 Quiz I Solutions

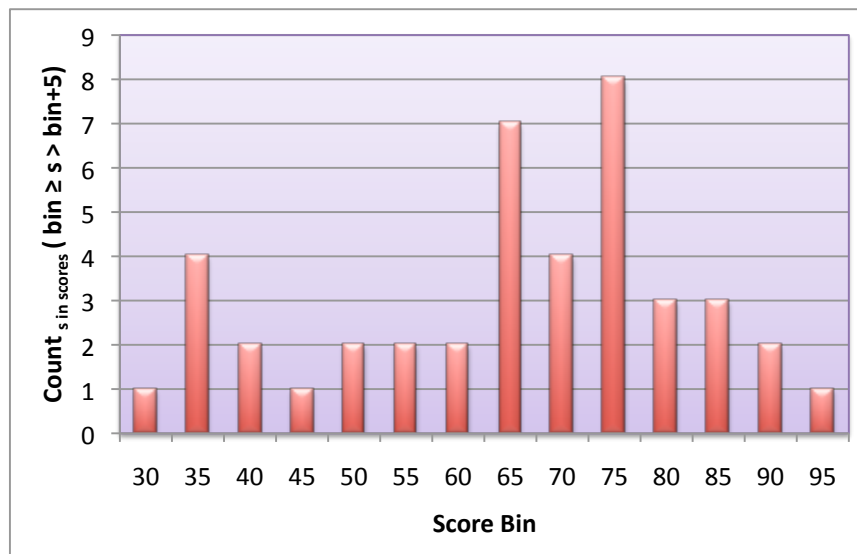
There are 15 questions and 12 pages in this quiz booklet. To receive credit for a question, answer it according to the instructions given. *You can receive partial credit on questions.* You have **80 minutes** to answer the questions.

Write your name on this cover sheet AND at the bottom of each page of this booklet.

Some questions may be harder than others. Attack them in the order that allows you to make the most progress. If you find a question ambiguous, be sure to write down any assumptions you make. Be neat. If we can't understand your answer, we can't give you credit!

**THIS IS AN OPEN BOOK, OPEN NOTES QUIZ.
NO PHONES, NO LAPTOPS, NO PDAS, ETC.**

Score Distribution



Name: **Solutions**

I Short Answer

1. [6 points]: Suppose the Selinger optimizer (described in the paper “Access Path Selection in a Relational Database Management System” by Selinger et al.) is being used to join four relations, A, B, C, and D. The table below shows the estimated costs and optimal orderings for the three-relation subplans. Here, the notation ABCD means a left-deep join of the four tables, with A as the lower-most, outer-most relation, e.g., (((A join B) join C) join D).

Costs are simply numbers generated by the optimizer; you aren’t expected to understand how they are derived or to use the cost model or selectivities from the Selinger paper in answering this question.

Subplan	Optimal Ordering	Cost
A,B,C	CAB	10
A,B,D	BAD	20
A,C,D	CAD	15
B,C,D	CBD	25

Given the above optimal three-relation joins, which of the following are orderings and costs that the optimizer *could* produce for the final A,B,C,D plan? Assume the costs of the plans in the table above do not tell you anything about the cost of the additional join required for this final plan.

(Circle ALL that apply.)

- A. CABD, cost 15
- B. CDAB, cost 30
- C. ACDB, cost 5
- D. BCAD, cost 20

Explanation: There are two things to note here: The final plan must have one of the current plans as its subplan and the total cost must be greater the cost of the current subplans. Choice A has CAB as its subplan and choice D has CAD as its subplan so both are possible final orderings.

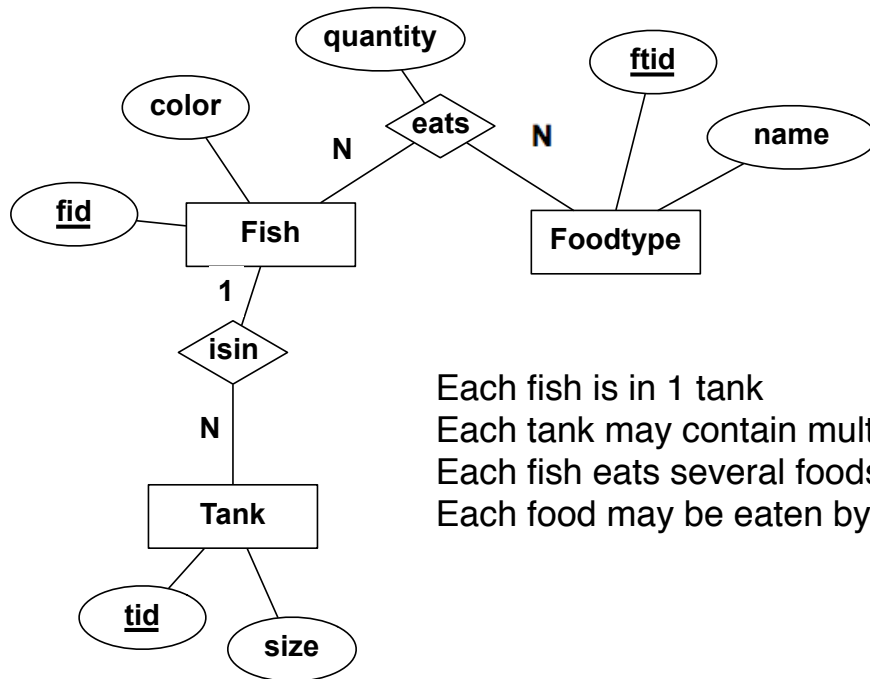
2. [6 points]: The paper “Join Processing in Database Systems with Large Main Memories”, by L.D. Shapiro, shows that the Grace hash-join generally outperforms sort-merge join. Assume that both join methods perform about the same amount of random I/O, that the tables are approximately the same size, and neither is initially ordered on the join attribute. Under what circumstances might a sort-merge join be preferable to Grace hash?

(Write your answer in 1–2 sentences in the space below.)

Answer: If we want the output to be sorted on the joined column because of an ORDER BY clause then using a sort-merge can be favorable. It can also be favorable if there are too many duplicates in the tuples so that performing a hash join will cause particular buckets to be too large.

II Fish Store Schema

You are given the following ER-diagram for a database that stores information about a fish store. Here the labels 1 and N on edges of the diagram indicate whether there is 1 entity or many (N) entities participating in a relationship.



Each fish is in 1 tank
 Each tank may contain multiple fish
 Each fish eats several foods
 Each food may be eaten by many fish

A hierarchical, or tree-structured schema is one in which each table has exactly one parent, unless it is a root table in which case it has zero parents (imagine an XML document without references to other nodes). For example, the following is one potential hierarchy that could be used to represent the fish store schema, along with some sample data:

Hierarchy:

```

fish
  color
  fid
  tank
    tid
    size
  foodtype
    ftid
    name
    quantity
    
```

Sample data:

```

<fish id=1 color=green>
  <tank id=1 size=50gal>
  <food id=1 name=flakes quantity=2>
  <food id=2 name=pellets quantity=3>
</fish>
<fish id=2 color=red>
  <tank id=2 size=25gal>
  <food id=1 name=flakes quantity=4>
</fish>
    
```

Observe that this schema has redundancy, since the same tank and foodtype information may be repeated in several fish records.

Suppose there are 1000 fish, 10 food types, and 100 tanks, that each fish is randomly and uniformly assigned food types and a tank, and that each fish eats on average two foods.

3. [8 points]: Devise a hierarchy to represent this data that has minimum redundancy (i.e., that will repeat the least information). Your answer should be similar to the hierarchy given above but with the tables nested differently.

(Write your answer in the space below.)

Answer: The schema below provides the minimum redundancy.

```
tank
  tid
  size
  fish
    fid
    color
    foodtype
      ftid
      name
      quantity
```

4. [8 points]: In addition to the functional dependencies suggested by the statements in the lower right of the diagram, assume that every attribute is functionally dependent on the key of the entity in which it appears. Write a collection of BCNF relations corresponding to this diagram.

(Write your answer in the space below.)

Answer:

fish: (fid, color, tank refs tank.tid)

fish_eats: (fid, ftid, quantity)

foodtype: (ftid, name)

tank: (tid, size)

Note that fish-tid is a many to one dependency, therefore we don't need an extra table for it.

III Query Planning

You are given the following schema and SQL query:

```
dept (did int primary key, bldg int, campus int) // 12 bytes per record
hobby (hid int primary key, hname char(17), cost int) //25 bytes per record
emp (eid int primary key, ename char(17), d int references dept.did) //25 bytes per record
hobbies (e int references emp.eid, h int references hobby.hid) //8 bytes per record
```

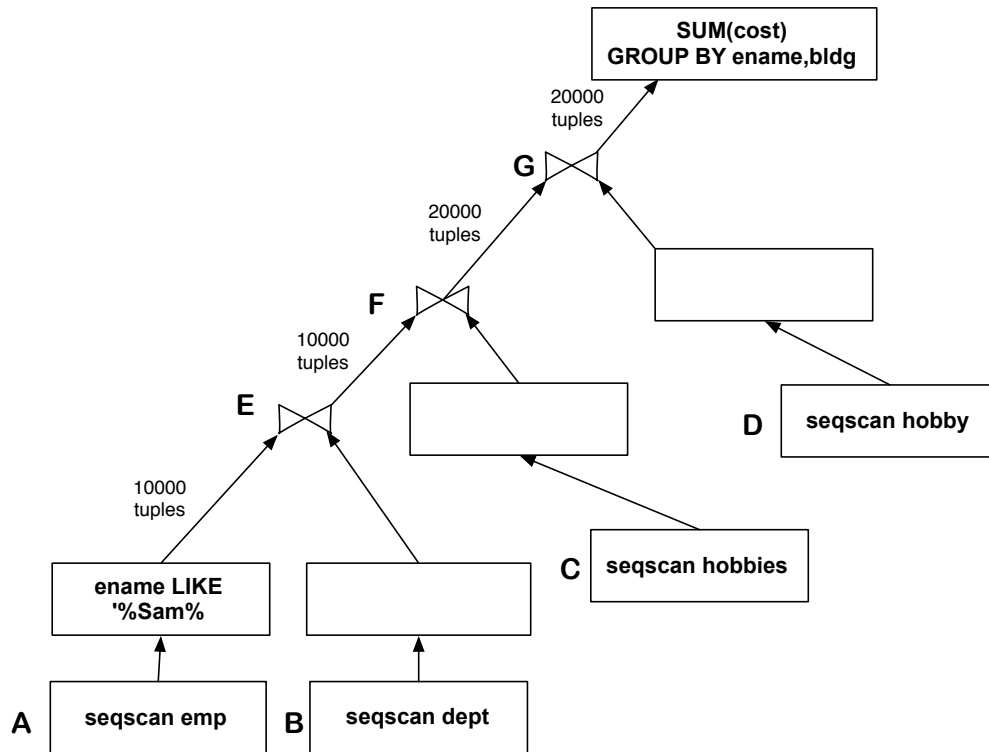
```
SELECT ename,bldg,SUM(cost)
FROM emp,dept,hobbies,hobby
WHERE emp.d = did
AND hobbies.e = eid
AND hobbies.h = hid
AND ename LIKE '%Sam%'
GROUP BY ename,bldg
```

You are given the following statistics (here, $|A|$ denotes the number of tuples in relation A , and $S(e)$ denotes the selectivity of expression e).

Statistic	Value
$ emp $	10^5
$ dept $	10^3
$ hobby $	10^3
$ hobbies $	2×10^5
$S(\text{ename LIKE '%Sam%'})$.1

Assume an integer is 4 bytes and a character is 1 byte, and that a disk page is 1000 bytes. Suppose you are running in a system with 100 pages of memory. Assume that each employee has about the same number of hobbies, and that hobbies and departments are assigned to employees uniformly and at random.

5. [10 points]: For now, assume that each join is a nested loops join and that there are no indices and no projection operations. In the diagram below, in the boxes labeled A, B, C, and D, write the names of the relations in the optimal left-deep join order for this plan. You should place the relation in the outer loop of the join in the leftmost box. Also indicate where the filters from the query should be placed. Some boxes may be empty. You should assume that the 100 pages of memory are optimally allocated between the joins to minimize the amount of I/O required by the plan.



Answer:

The hobby and dept tables can both fit into RAM. The emp and hobbies do not. By doing the emp-dept join first, with emp as the outer, we are able to scan emp just once. We only scan dept once because it fits into memory. We have to do the join with hobbies next (because the join with hobby would be a cross product). Due to the constraints of left-deep plans, we must put the hobbies table on the inner, requiring us to scan it once for each tuple output by the emp/dept join (10^4 times). Finally, we can do the join with hobby; since hobby fits into memory, we only scan it once as well.

The following table summarizes the pages used by each of the tables:

Table	Formula	No. pages
emp	$10^5 / (1000 / 25)$	2500
dept	$10^3 / (1000 / 12)$	12
hobby	$10^3 / (1000 / 25)$	25
hobbies	$2 \times 10^5 / (1000 / 8)$	1600

6. [6 points]: Estimate the total number of disk pages read by the plan you drew above (do not worry about seeks for this problem).

(Write your answer in the space below.)

Answer: The total I/O cost is one scan of dept + one scan of hobby + one scan of emp + 10^4 scans of hobbies. Emp is 2500 pages, dept is 12 pages, and hobbies is 25 pages. Summing these numbers, we get: $1.6 \times 10^7 + 2500 + 12 + 25 \approx 1.6 \times 10^7$ pages

7. [6 points]: Estimate the number of tuples produced by the plan you drew above.

(Write your answer in the space below.)

The top-most join produces 20,000 records, but this represents only 10,000 distinct employees, so the GROUP BY will output 10,000 values (assuming employee name are unique.)

8. [6 points]: Suppose that the system runs for awhile and the database grows to have 10 times as much data in every table.

Which of the following would likely improve the runtime of the above query on this larger database by more than a factor of 2, assuming its total runtime is dominated by I/O, that seeks take about $100\times$ longer than the time to read a page from disk, and that the optimizer makes optimal decisions about join ordering and index selection.

(Circle ALL that apply.)

- A. Create an unclustered B+Tree on emp.ename
- B. Use a Grace Hash Join for the join between employees and departments
- C. Use a Grace Hash Join for the join between employees and hobbies
- D. Create an unclustered hash index on emp.ename

Answer: Indices won't help because the predicates are not very selective, and so indices will result in a lot of random I/O. Grace hash on employees and hobbies is a win because this join dominates the overall I/O cost as a nested loops, and grace hash will do far less I/O. Grace hash between employees and departments won't help because the departments table fits into RAM, so a nested loops join only scans both tables once, whereas a grace hash scans them multiple times.

IV Bitmap Indices

Dana Bass proposes a new kind of index called a “bitmap index” (note that these are different than the bitmaps you may have seen in some of the query plans generated in PS2 by Postgres). Like other indices, it is built on a single column of a table. It contains one bitmap for each distinct value that appears in that column. For example, a bitmap index on gender would contain two bitmaps, one for “M” and one for “F”; the column “MMMFFF” would be indexed into the bitmaps “M: 111000” and “F: 000111”. Each bitmap requires as many bits as there are rows in the table in memory or on disk to store. Such bitmaps can be used in place of sequential scans for computing many filter and aggregate queries – the idea is that for certain queries, the database can simply look at the bits that are set in one or more of the bitmaps and determine which tuples satisfy the query and / or compute the values of aggregates (e.g., COUNTs) without consulting the heap file at all.

Dana is administering the database for a large insurance company, and is running aggregate queries to count the frequencies of certain pre-existing conditions amongst the company’s insurees. The database has one row per insuree in the insurees table. Each patient has exactly one condition, and the id of that condition is recorded in the table. There are 100 distinct conditions ranging from 1 . . . 100. She runs the query:

```
SELECT condition, COUNT(*)
FROM insurees
GROUP BY condition
```

9. [8 points]: Dana finds that it takes N minutes to run this query via a sequential scan of the heap file, which contains 100 byte tuples. Assuming this query is completely dominated by I/O cost, that none of the bitmaps or the table are in memory before the query is run, and that the table is many millions of rows, estimate how long it would take to run this query with a bitmap-index on condition. Explain your answer in 1–2 sentences.

(Write your answer in the space below.)

Answer: $\frac{N}{8}$

For this query, you need to scan 100 bitmaps in total. Each bitmap uses 1 bit per tuple, rather than 100 bytes. A database for a large insurance company is likely to be quite large, so assume that the extra 100 seeks required by the bitmap query require negligible time (or that the bitmaps are stored sequentially on disk, so that no additional seeks are needed). So, multiply the time spent in the full query by the ratio of bytes read with the indices to bytes read in a sequential scan: $N \cdot \frac{100 \text{ bitmaps} \cdot \frac{1}{8} \frac{\text{bytes}}{\text{bitmap-tuple}} \cdot \text{Ktuples}}{100 \frac{\text{bytes}}{\text{tuple}} \cdot \text{Ktuples}} = \frac{N}{8}$.

V Join Algorithms

You have a database with two tables, T1 and T2. Both are many times larger than memory, and contain millions of rows, each with multiple columns. You'd like to execute an equality join of T1.C1 with T2.C2, with each tuple of output including all columns from both tables. T1.C1 and T2.C2 are both keys for their tables (that is, neither table has duplicate values in its column). Each T1.C1 value occurs once in T2.C2. Both tables are organized as heap files, in no particular order.

For each question below, you should name the best join algorithm for the situation described. Here are some possibilities, though you can use other algorithms if they are better.

- Nested loops join.
- GRACE hash join.
- Sort heap files, then merge-join.
- Merge-join on B+Trees (without sorting).
- Scan T1, look up in T2's B+Tree (index nested-loops join).

10. [6 points]: T2 has a B+Tree index on its join column, T2.C2. The index is many times larger than memory. What is the fastest join algorithm, and why?

(Write your answer in the space below.)

Answer: GRACE hash joins are the fastest. Sorting the heap files then doing a merge join, has comparable IO complexity; but as argued by the paper on GRACE hash joins, it has substantially higher CPU costs. The B+Tree indices are not very useful for these queries; because they don't fit into memory and you're selecting out all rows from each table, you will spend a huge amount of time doing disk seeks if you try to use the B+Tree.

11. [6 points]: Each table has a B+Tree index on its join column; both indices are many times larger than memory. What is the fastest join algorithm, and why?

(Write your answer in the space below.)

Answer: GRACE hash joins are the fastest. See above for reasoning regarding sort / merge join and using the B+Trees. The situation with the B+Trees will be even worse if you try to use both trees at once, as you won't even be reading one table sequentially.

12. [6 points]: You add a predicate on T1 to the query; the predicate is true for only 10 rows. The predicate does not use T1.C1. Again, each table has a B+Tree index on its join column. What is the fastest join algorithm, and why?

(Write your answer in the space below.)

Answer: Scan T1, look up in T2's B+Tree (index nested-loops join) is the fastest option. The filter predicate sharply reduces the number of rows that are output by T1; doing a small handful of disk seeks to pull those 10 tuples out of T2 is going to be much faster than sequentially scanning all of T2 given that T2 is much larger than main memory.

VI Serializability

You're running the reservation system for an event that will take place in a concert hall with 100 seats. Your application has just one requirement, which we'll call "consistency:" that no two customers get the same seat. You define a database with two tables. The `seats` table contains a row for each seat; each row has a `id` field, which is an integer between 1 and 100, and a `taken` field, which is a boolean that indicates whether the seat is already reserved. Your database also has a `hint` table with just one row; the row has just one entry, called `x`, which is an integer.

You define the following transaction:

```
function reserve():
    begin transaction
    L1. h = select x from hint;
        while(h < 100){
    L2.   t = select taken from seats where id = h;
        if(t == False){
    L3.   update seats set taken = True where id = h;
        print h;
        break; // go to L4
        }
        h = h + 1;
    }
    L4. update hint set x = h + 1;
        commit
```

The `Lx` are labels for use in describing schedules; they are not part of the code. With respect to this code, consistency means that no two calls to `reserve()` print the same seat number.

The initial database contents are as follows. The `seats` table has 100 rows, with IDs 1..100, all with `taken=False`. The `hint` table has one row with `x = 1`.

For each of the following schedules, indicate whether it is conflict-serializable, whether it is serializable, and whether it would leave the database in a consistent state as defined above by the application. Each schedule starts with the initial database contents given above.

13. [6 points]:

T1	T2
--	--
L1	
L2	
L3	
	L1
	L2
	L2
	L3
L4	
	L4

Circle YES or NO for each item.

- A. Conflict-serializable? No. T2 reads seats row 1 after T1 modifies it, but T2 also reads the hint row after T1 modifies it. These conflicts prevent us from rearranging the schedule into either serial order of T1 and T2.
- B. Serializable? Yes. The final database contents, and printed output, are the same as the serial order T1 then T2.
- C. Consistent? Yes. The two transactions printed different seat numbers, and marked different seats as taken.

14. [6 points]:

T1	T2
--	--
L1	
L2	
	L1
	L2
L3	
L4	
	L3
	L4

Circle YES or NO for each item.

- A. Conflict-serializable? No. T2 reads seats row 1 after T1 writes it, but reads hints before T1 modifies it. These conflicts prevent us from rearranging the schedule into either serial order.
- B. Serializable? No. The final database contents have just seat 1 marked taken and the hint set to 2, and both transactions print "1". Either serial order would have marked two seat rows taken, and printed "1" and "2".
- C. Consistent? No. Two customers have been given the same seat.

15. [6 points]:

T1	T2
--	--
L1	
L2	
L3	
	L1
	L2
	L2
	L3
	L4
L4	

Circle YES or NO for each item.

- A. Conflict-serializable? No. T2 reads seats rows 1 after T1 updates it, but T2 writes hints before T1 writes hints. These conflicts prevent us from rearranging the schedule into either serial order of T1 and T2.
- B. Serializable? No. The final hint value is 2, while either serial order would yield a hint value of 3.
- C. Consistent? Yes. The two customers have been given different seats.

End of Quiz I