Department of Electrical Engineering and Computer Science

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6.5830/31 Database Systems: Fall 2022 Quiz I

There are 23 questions and 17 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.
LAPTOPS MAY BE USED; NO PHONES OR INTERNET ALLOWED.

Do not write in the boxes below

1-7 (xx/32)	8-14 (xx/24)	15-20 (xx/28)	21-23 (xx/16)	Total (xx/100)

Name:

I SQL

Suppose you have a table of employees recording their salaries and the department they work in, a table of departments and the buildings they are in, and a table recording which employees manage which other employees.

```
emp (id, name, dno)
dept (did, bldg)
manages (mgrid, eid)
```

Suppose that:

- There are exactly 100 employees, with ids from 1 to 100
- There are at least 5 departments
- `dno` is a foreign key reference to `did`
- `mgrid` and `eid` are foreign key references to `id`
- There are employees in at least departments 1, 2 and 3
- Every employee has at least one manager
- Employee id 1 is a manager (i.e., appears as a `mgrid` in `manages`)

Consider the following 3 SQL queries, listed with 6 possible outputs. For each query and output, circle the outputs that could have possibly resulted from the query, given the constraints above.

1. [6 points]: What are the possible outputs of this query?

```
SELECT COUNT(DISTINCT dept.did)
FROM dept JOIN emp ON dno=did
WHERE did = 4
```

(Circle all possible outputs.)

A. 0

B. 1

C. 2

D. 3

E. 100

F. 101

Answer: A B

Name:

2. [6 points]: What are the possible outputs of this query?

```
SELECT COUNT(*)
FROM dept RIGHT JOIN emp ON dno=did
```

(Circle all possible outputs.)

A. 0

B. 1

C. 2

D. 3

E. 100

F. 101

Answer: E

3. [6 points]: What are the possible outputs of this query?

```
WITH mgr_depts AS (
 SELECT mgrid, did
 FROM manages JOIN emp ON eid = id
 JOIN dept ON did=dno
)
SELECT count(*) FROM mgr_depts
WHERE did = 1 and mgrid = 1
```

(Circle all possible outputs.)

A. 0

B. 1

C. 2

D. 3

E. 100

F. 101

Answer: A B C D

Name:

II Relational Model

Consider the tables below; assume these are the complete contents of these tables.

```
Table A              Table B
sam    ram  ham      mike  like   bike
1      a    f        1     a      f
2      b    g        1     b      f
3      c    k        2     b      g
```

4. [4 points]: Based on this data, answer true/false for each of the following:
(Circle 'T' or 'F' for each choice.)

T **F** `sam` is definitely the primary key for Table A

T **F** `ham` could be a foreign key reference to `B.bike`

T **F** `like` could be a foreign key reference to `A.ram`

T **F** `mike` is definitely not a primary key for Table B

Answer: F F T T

Name:

III Column Stores

Consider the following query over a table with 3 integer columns, col1, col2, and col3.

```
SELECT COUNT(distinct col1)
FROM table
WHERE col2 = 1
```

Suppose the table is sorted on `col2` and then `col1`, and that `col1` is an integer between 1 and 10,000 chosen at random and that `col2` is an integer that is either 0 or 1 with 50% of values being 0 and 50% being 1. There are no indexes. The query executor has one access method – a bitmap sequential scan – which takes as input a bitmap indicating the records to read and reads the marked records in the bitmap. For scans of tables where bitmaps are not available all records are scanned (i.e., the bitmap is assumed to be all 1s.)

The table has 1 million rows, and you have a disk that can read 1 MB/sec. Integers are 8 bytes. Ignoring CPU costs, estimate the time to answer this query under the following settings:

5. [3 points]: An uncompressed row store.

(Circle the best answer.)

A. < 2 sec

B. 4 sec

C. 8 sec

D. 12 sec

E. 16 sec

F. > 20 sec

Answer: F. There are three columns, and each takes 8 seconds to read.

Name:

6. [3 points]: An uncompressed late materialization column store.

(Circle the best answer.)

A. < 2 sec

B. 4 sec

C. 8 sec

D. 12 sec

E. 16 sec

F. > 20 sec

Answer: D. It takes 8 seconds to read col2 and produce a bitmap on it, which selects 50% of the values that are stored consecutively. This bitmap is used to select half of col1, which takes 4 seconds.

7. [4 points]: A late materialization column store where `col2` is compressed with RLE and `col1` is uncompressed.

(Circle the best answer.)

A. < 2 sec

B. 4 sec

C. 8 sec

D. 12 sec

E. 16 sec

F. > 20 sec

Answer: B. It takes because col2 is RLE encoded, it occupies very little space (just 2 runs), so it takes close to 0 seconds to read and produce a bitmap with 50% of the values set to 1. This bitmap is used to select half of col1, which takes 4 seconds.

Name:

IV Postgres Plan Interpretation

Consider the two following Postgres commands:

```
EXPLAIN ANALYZE SELECT s.station_id FROM stations s,
rail_ridership rr WHERE s.station_id = rr.station_id AND
s.station_id ILIKE '%x%';

EXPLAIN ANALYZE SELECT s.station_id FROM stations s,
rail_ridership rr WHERE s.station_id = rr.station_id AND
s.station_id ILIKE '%a%';
```

When you run them in Postgres, you get the following output:

```
                                QUERY PLAN 1
-----------------------------------------------------------------------------------------------------
 Nested Loop  (cost=0.28..216.67 rows=65 width=11)
              (actual time=0.126..0.430 rows=132 loops=1)
   ->  Seq Scan on stations s  (cost=0.00..2.50 rows=1 width=11)
                               (actual time=0.095..0.234 rows=1 loops=1)
       Filter: (station_id ~~* '%x%'::text)
       Rows Removed by Filter: 119
   ->  Index Only Scan using station_id on rail_ridership rr  (cost=0.28..213.47 rows=70 width=11)
                               (actual time=0.027..0.162 rows=132 loops=1)
       Index Cond: (station_id = s.station_id)
       Heap Fetches: 132
 Planning Time: 0.947 ms
 Execution Time: 0.480 ms

                                  QUERY PLAN 2
-----------------------------------------------------------------------------------------------------
 Hash Join  (cost=4.00..202.88 rows=7854 width=11)
            (actual time=0.343..7.580 rows=7854 loops=1)
   Hash Cond: (rr.station_id = s.station_id)
   ->  Seq Scan on rail_ridership rr  (cost=0.00..177.54 rows=7854 width=11)
                                  (actual time=0.013..1.863 rows=7854 loops=1)
   ->  Hash  (cost=2.50..2.50 rows=120 width=11)
                                  (actual time=0.315..0.316 rows=120 loops=1)
         Buckets: 1024  Batches: 1  Memory Usage: 14kB
         ->  Seq Scan on stations s  (cost=0.00..2.50 rows=120 width=11)
                                (actual time=0.020..0.250 rows=120 loops=1)
               Filter: (station_id ~~* '%a%'::text)
 Planning Time: 1.008 ms
 Execution Time: 8.380 ms
```

Name:

8. [2 points]: In query plan 1, which join algorithm does Postgres use?
(Circle the best answer)

A. Nested loops

B. In-memory hash join

C. Index nested loops

D. Bitmap heap join

E. Grace hash join

F. None of the above
Answer: C

9. [2 points]: In query plan 2, which join algorithm does Postgres use?
(Circle the best answer)

A. Nested loops

B. In-memory hash join

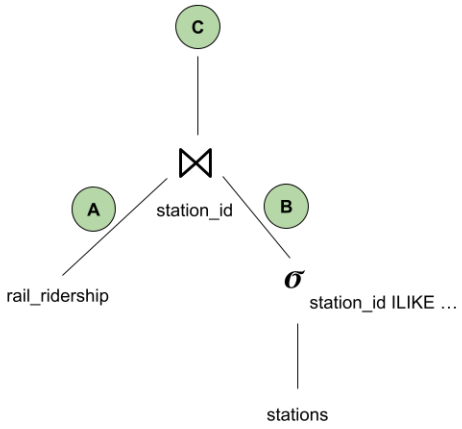
C. Index nested loops

D. Bitmap heap join

E. Grace hash join

F. None of the above
Answer: B

The following two questions refer to the figure below, which roughly depicts the structure of both query plans.



10. [4 points]: For query plan 1, fill in both the estimated and actual cardinalities at each spot in the figure above.

A. Estimated: _____ Actual: _____

B. Estimated: _____ Actual: _____

C. Estimated: _____ Actual: _____

Answer: A. Estimated: 70, Actual: 132 B. Estimated: 1, Actual: 1 C. Estimated: 65, Actual: 132

11. [4 points]: For query plan 2, fill in both the estimated and actual cardinalities at each spot in the figure above.

A. Estimated: _____ Actual: _____

B. Estimated: _____ Actual: _____

C. Estimated: _____ Actual: _____

Answer: A. Estimated: 7854, Actual: 7854 B. Estimated: 120, Actual: 120 C. Estimated: 7854, Actual: 7854

12. [3 points]: Notice that in both queries, we are joining the same two tables. Why do the two query plans use different join algorithms?

(Circle all that apply.)

A. The nested loop join avoids building a hash table.

B. Hash joins are better than nested loop joins; Postgres made a query planning error here.

Name:

C. The relative sizes of the two tables (which one is smaller / larger) switches between the two plans.

D. Postgres is substantially mis-estimating the number of stations with 'x' in their name.
Answer: A

Name:

The remaining questions in this section refer to the following scenario.

Suppose there is a table with columns `a`, `b`, `c`, and `d`. Little Bobby Tables is experimenting with indexes, so he adds an index on `(b, a)` to the table. He then runs the following query

```
SELECT ... FROM table WHERE a > N AND b = M
```

He is disappointed to discover that Postgres chooses not to use the index, opting for a Sequential Scan instead.

13. [5 points]: What are possible reasons why this might happen?
(Circle all that apply.)

A. Postgres estimates that 90% of the tuples in the table are selected by the filter.

B. The index is not clustered, and Postgres estimates the selectivity of the filter to be 50%.

C. Postgres estimates that less than 1% of the tuples in the table are selected by the filter, so an Index Scan is not needed for so few tuples.

D. Since the index is sorted on column `b`, then `a`, the index could not possibly accelerate this query.

E. The index can only help accelerate queries of the form `SELECT ... FROM table WHERE b > M`.
Answer: A B

14. [4 points]: Frustrated, Bobby decides to ask you for help on the following queries. Select true if there exists a scenario where Postgres can use the index on `(b, a)` to possibly accelerate the query.
(Circle 'T' or 'F' for each choice.)

T **F** `SELECT ... FROM table WHERE a = N`

T **F** `SELECT ... FROM table WHERE b = M`

T **F** `SELECT ... FROM table where a > N`

T **F** `SELECT ... FROM table where b > M`
Answer: F T F T

Name:

V Query Plans and Cost Models

Consider a database management system with:

- 50 tuples/page
- 20 KB/page
- Magnetic storage that provides:
  - 10 ms / random access page
  - 100 MB/sec sequential bandwidth

This system manages a manufacturing database with the following schema and data statistics:

```
part(partid, partname, partprice, weight, prodid)
product(prodid, prodname, prodprice, mfgid)
manufacturer(mfgid, mfgname, wid)
warehouse(wid, location)
```

```
|warehouse| = 50 records = 1 page
|manufacturer| = 300 records = 6 pages
|product| = 50,000 records = 1,000 pages
|part| = 1,000,000 records = 20,000 pages
```

For the next few questions, imagine a user wants to list all parts that cost under 10 dollars and which are included in products produced by a particular manufacturer:

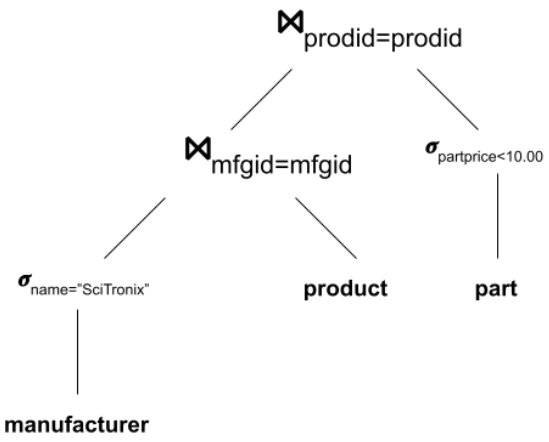
```
SELECT partid, partname
FROM part, product, manufacturer
WHERE part.prodid = product.prodid
AND       product.mfgid = manufacturer.mfgid
AND       manufacturer.mfgname = "SciTronix"
AND       part.partprice < 10.00
```

And has data such that:

- Of the tuples in the manufacturer table, there is exactly one with the name “SciTronix”
- That manufacturer makes 200 unique products
- The selectivity of $\sigma_{\texttt{partprice<10.00}}$ is 20%

Name:

This query yields the following plan:



15. [4 points]: What is the cost (in time) of the subplan that includes $\bowtie_{\texttt{mfgid=mfgid}}$? The plan uses heap scans. Account only for I/O operations and ignore CPU costs.

(Write your answer in the space below.)

*Answer: Cost is the sum of reading each of manufacturer and product. Manufacturer: 1 seek (which costs 0.01 seconds), plus scan of 6 pages. That's 6 * 20KB, for a total of 120KB scanned. At 100MB/sec, that's 0.0012 seconds for the scan. So 0.0112 seconds for manufacturer. Product: 1 seek (which costs 0.01 seconds), plus scan of 1000 pages. That's 1000 * 20KB, for a total of 20MB scanned. At 100MB/sec, that's 0.2 seconds for the scan. So 0.21 seconds for product. That's a sum of 0.2212 seconds for this subplan.*

16. [6 points]: What is the cost (in time) of the subplan that includes the $\sigma_{\texttt{partprice<10.00}}$ operator if we use a heap scan? Account only for I/O operations and ignore CPU costs.

(Write your answer in the space below.)

*Answer: Cost is: 1 seek (which costs 0.01 seconds), plus scan of 20K pages. That's 20K * 20KB, for a total of 400MB scanned. At 100MB/sec, that's 4 seconds for the scan. So 4.01 seconds.*

17. [6 points]: What is the cost of that subplan if we use a clustered B+Tree index on `part's partprice` attribute? The B+Tree's internal nodes all fit in memory.

(Circle the best answer.)

Name:

A. 0.02 seconds

B. 0.82 seconds

C. 40.02 seconds

D. 400.02 seconds

Answer: B

Name:

Now, consider four kinds of indexes:

A. B+Tree, clustered
B. B+Tree, unclustered
C. Hash Index
D. R-Tree

What is the most appropriate index for the following queries? Name the table, the attribute(s), and the index chosen from the above list. In some cases, multiple index types might be suitable; in that case, list one. Assume for each query that this is the only query run in the database.

18. [4 points]: `SELECT MAX(prodprice) FROM product`

(Write the best index(es) in the space below. Include the table, attribute(s), and index type.)

Answer: Clustered or unclustered B+Tree product(prodprice)

19. [4 points]: `SELECT * FROM part WHERE partprice > 30`

(Write the best index(es) in the space below. Include the table, attribute(s), and index type.)

Answer: Clustered B+Tree part(partprice).

20. [4 points]: `SELECT COUNT(*) FROM part WHERE partid = 10`

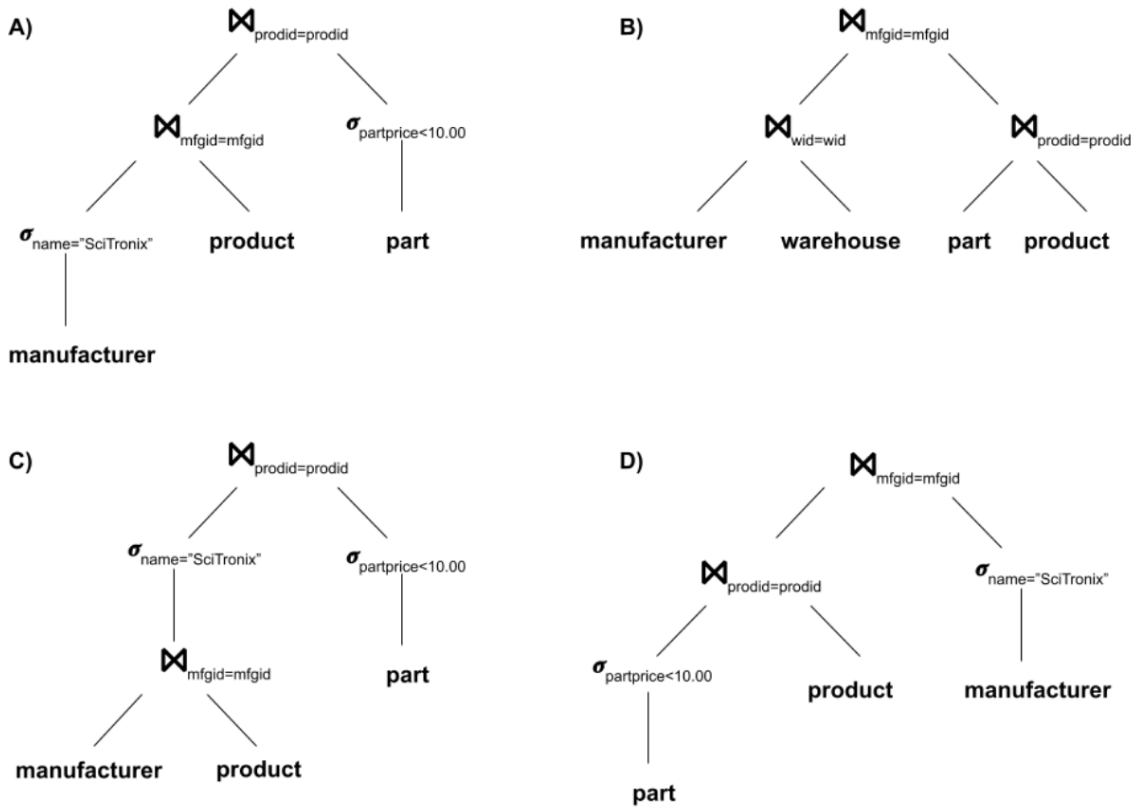
(Write the best index(es) in the space below. Include the table, attribute(s), and index type.)

Answer: Hash index part(partid).

Name:

VI Query Optimization

The following plans reflect several different possible queries over the schema described in the previous section. Some plans may reflect the same query.



21. [8 points]: Which of the above plans might be evaluated for cost by the Selinger optimizer? Choose zero or more answers.

(Circle all that apply.)

A. Plan A

B. Plan B

C. Plan C

D. Plan D

Answer: Plan A is fine. Plan B is non-left-deep, so the Selinger optimizer would not consider it. Plan C has not pushed predicates, so this would never be costed. Plan D is fine.

Name:

VII Joins

Consider a join in which the outer table X has 100 pages. The inner table Y has 200 pages. We have 50 pages of RAM available. Both inputs are base tables.

22. [4 points]: How many pages are read when executing a nested loop join in this scenario?
(Write your answer in the space below.)

Answer: $100 + \{X\} \times 200$

23. [4 points]: How many pages are read when executing a blocked nested loop join, if the block size B is 25 pages?
(Write your answer in the space below.)

Answer: 900

End of Quiz I!

Name: