

**⚠ This quiz has been regraded; your score was not affected.**

# Final Exam

**Due Dec 16 at 5:15pm      Points 100      Questions 15**

**Available** Dec 16 at 2:45pm - Dec 16 at 5:15pm about 3 hours

**Time Limit** 150 Minutes

This quiz was locked Dec 16 at 5:15pm.

## Attempt History

|               | <b>Attempt</b>                   | <b>Time</b>    | <b>Score</b>       | <b>Regraded</b>    |
|---------------|----------------------------------|----------------|--------------------|--------------------|
| <b>KEPT</b>   | <a href="#"><u>Attempt 2</u></a> | 132 minutes    | 93 out of 100      | 93 out of 100      |
| <b>LATEST</b> | <a href="#"><u>Attempt 2</u></a> | 132 minutes    | 93 out of 100      | 93 out of 100      |
|               | <a href="#"><u>Attempt 1</u></a> | 75,047 minutes | 21.67 out of 100 * | 21.67 out of 100 * |

\* Some questions not yet graded

**⚠ Correct answers are hidden.**

Score for this quiz: **93** out of 100

Submitted Dec 16 at 5:15pm

This attempt took 132 minutes.

| <b>Question 1</b> | <b>10 / 10 pts</b> |
|-------------------|--------------------|
|                   |                    |

Consider the following database schema about books and user ratings:

- **Book (bid, title, author, genre)**. The primary key is **(bid)**.
- **Rating (uid, bid, rating)**. The primary key is **(uid, bid)**. There is also a foreign key dependence from **Rating.bid** to **Book.bid**.

Write a SQL query that outputs, for every author that has written at least 3 books in the "Fiction" genre, **the average rating** of all their books in decreasing order of average rating.

Your Answer:

```
WITH AuthorsFictionNumbers(author, fiction_number) AS (
    SELECT b.author AS author, COUNT(*) AS fiction_number
    FROM Book b
    WHERE b.genre = "Fiction"
    GROUP BY b.author
), Authors(author) AS (
    SELECT a.author AS author
    FROM AuthorsFictionNumbers a
    WHERE a.fiction_number >= 3
)
SELECT r.author, AVG(r.rating) AS avgRating
FROM Book b, Rating r, Authors r
WHERE b.bid = u.bid AND b.author = r.author
GROUP BY r.author
ORDER BY avgRating DESC;
```

**Question 2****9 / 9 pts**

Consider a relational schema with two relations  $R(A, B)$ ,  $S(B, A, C)$ , and the following query in Relational Algebra:

$$q = \pi_A(R) \cup \pi_A(\sigma_{C=5}(S))$$

Which of the following queries are equivalent to  $q$ ? Choose all the correct options.

$\pi_A(R \cup \sigma_{C=5}(S))$

$\pi_A(R) \cup \sigma_{C=5}(\pi_A(S))$

$\pi_A(R \cup \pi_{A,B}(\sigma_{C=5}(S)))$

**Question 3****10 / 10 pts**

Consider the relation  $R(A, B, C, D, E)$  with functional dependencies:

$A, B \rightarrow C$

$C \rightarrow A$

$D \rightarrow E$

Choose the all correct options:

- ABD is a key in R
- CD is a superkey in R
- R is in BCNF
- R is in 3NF
- The set of FDs is a minimal basis.

#### Question 4

8 / 8 pts

We are given a relation R(A, B) with **80 pages** and a relation S(C, D, E) with **50 pages**. Suppose that A is the primary key in R, C is the primary key in S, and there is a foreign key dependency from R.B to S.C.

Each of the attributes A, B, C, D, E has **20 bytes**. Moreover, attribute A takes values uniformly in the range [1,100].

How many pages do we need to store the output of the following SQL query? Assume that the size of a page is **1,000 bytes**.

```
SELECT R.A, S.D, S.E  
FROM R, S  
WHERE R.B = S.C AND R.A <= 10
```

12

**Question 5****5 / 5 pts**

Explain your answer to the above question.

Your Answer:

(Question 4&5:

Each attribute has 20 bytes  $\rightarrow$  Tuple in R has 40 bytes, tuple in S has 6

$A \in [1..100] \rightarrow$  Selectivity of  $R.A \leq 10 = 10/100 = 10\%$

$R : 80 \text{ pages} * 1000 \text{ bytes/page} / 40 \text{ bytes/tuple} = 2000 \text{ tuples}$

After selection,  $2000 * 10\% = 200$  tuples are left.

$S : 50 \text{ pages} * 1000 \text{ bytes/page} / 60 \text{ bytes/tuple} = 2500/3 \text{ tuples}$

After the join, as there's foreign key dependency from R.B to S.C, the result 200 tuples.

Each page has 3 attributes  $\rightarrow$  each output tuple is 60 bytes.

$\text{Result} = 200 \text{ tuples} * 60 \text{ bytes/tuple} / 1000 \text{ bytes/page} = 12 \text{ pages}$

**Question 6****6 / 6 pts**

We are given two relations: R with 30,000 pages and S with 10,000 pages. We are performing a key-foreign key join between R and S, wherein S has the foreign key attribute. Suppose that R is already sorted on the join attribute.

**What is the I/O cost of the Sort Merge Join algorithm that uses replacement sort to create the initial runs? Do not count the cost of writing the join result to disk. Assume that the size of the buffer is 100 pages.**

60000

### Question 7

5 / 6 pts

Explain your answer to the above question.

Your Answer:

Question 6 & 7:

The I/O cost of SMJ  $\geq \text{sort}(R) + \text{sort}(S) + M_r + 1$

$R$  is already sorted  $\rightarrow \text{sort}(R) = 0$

Sort  $S$ , I/O = 1 pass  $= 2 * 10000 = 20000$

Add them up, Ans  $= 0 + 20000 + 30000 + 10000 = 60000$

We do not need to sort  $S$  totally, you can create initial runs and in the merge stage do the join simultaneously.

## Question 8

4 / 4 pts

We want to compute **the cartesian product  $R \times S$**  of two relations.  
Which of the following join algorithms are appropriate for this task?

- Block Nested Loop Join
- Hash Join
- Sort Merge Join
- Nested Loop Join

**Question 9****9 / 15 pts**

Suppose we have a DB schema with one relation:

- R(A,B,C,D) has 1,000,000 tuples.

Consider the following SQL query:

```
SELECT A, SUM(D)
FROM R
WHERE R.C = 10
GROUP BY A;
```

Assume that:

- Each page holds 1,000 tuples
- The selectivity of the predicate ( $R.C = 10$ ) is 10%
- The buffer has size 20 pages.
- R is already sorted on attribute A

**What is the I/O cost of the best query plan? Ignore the cost of writing the result to disk.** Describe your answer in detail by presenting the annotated query plan (Here you just need to describe the sequence and implementation of each operator, and where you use materialization or pipelining).

Your Answer:

The I/O cost is 200 I/Os.

R has  $1000000/1000 = 1000$  pages.

Firstly do a selection on R ( $R.C = 10$ ) using a hash index. As the selectivity of the predicate is 10%, the intermediate result contains 100 pages. Here, as we are using a hash index, we only need 100 I/Os to read the pages. The result from the first step has 100 pages, which does not fit in a buffer with 20 pages, so the intermediate result is materialized.

Then, we do an aggregation on A and sum the values of D using the result in the previous step. We do know that R is already sorted on attribute A, so we only need to go through all entries and compare the value of A to see if the D affiliated with A should be summed up.

Adding them up and ignoring the cost of writing the result to disk, this plan has the cost of  $100 + 100 = 200$  I/Os.

We're not assuming any index. What about the cost of reading in records initially? Selection can be pipelined.

### Question 10

4 / 4 pts

Consider an extendible hashing index with a directory that has 16 entries. Suppose that each page in the index can hold up to 10 data entries. We want to calculate the smallest and largest number of data entries that this index can hold.

- smallest:

- largest:

---

**Answer 1:**

11

---

**Answer 2:**

160

**Question 11**

4 / 4 pts

Consider a relation R(A, B, C, D). We want to build a **B+ tree index** so that we can speed up queries that can take one of the following forms:

```
SELECT *
FROM R
WHERE R.A = ? AND R.B = ? ;
```

```
SELECT *
FROM R
WHERE R.B = ? ;
```

Here, ? can take different constant values. **What should we choose as the search key for the index?** Explain your answer in detail.

Your Answer:

We should be able to choose a subset in the predicates of either query (1) or (2) and form a prefix of our search key. So, the first key in our search key has to be B. We can then append A, as the search key B+ (B, A) works for both queries (query 1: Select B, A. query 2: Select B.)

## Question 12

3 / 3 pts

Consider the following SQL query:

```
SELECT *
FROM R
WHERE R.A = 1 AND NOT (R.B = 0) AND R.C > 10;
```

A **hash index on (B)** matches the selection predicate in the above query.

True False**Question 13****4 / 4 pts**

A protocol that writes a page to disk immediately after the page is modified by a transaction guarantees atomicity.

 True False**Question 14****4 / 4 pts**

If a transaction reads a data object after it has been written by an uncommitted transaction, then isolation is always violated.

 True False**Question 15** Original Score: 8 / 8 pts Regraded Score: 8 / 8 pts

❗ This question has been regraded.

Consider three transactions T1, T2, T3, and the interleaved schedule:

**R1(A), R3(C), W1(A), R2(B), W3(C), R3(D), R2(A), W2(A), R1(E).**

The above schedule is conflict equivalent to the following serial schedules:

- T3, T1, T2
- T2, T1, T3
- T1, T3, T2
- T2, T3, T1

Quiz Score: **93** out of 100