SWE3003 Introduction to Database Systems - Midterm Fall 2023

Student ID	Name

For Instructor/TA only,

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Total

Academic Honor Pledge

I affirm that I will not at any time be involved with cheating or plagiarism while enrolled as a student Programming Language class at SungKyunKwan University. I understand that violation of this code will result in penalties as severe as indefinite suspension from the university.

Your signature: ____________________

1. [30 pts] For each of the following statements, indicate whether it is TRUE or FALSE. You will get 3 points for each correct answer, -3 points for each incorrect answer, and 0 point for each answer left blank or both answers marked.

		T	F
(a) Each superkey is a superset of some candidate key	T	☐	☐
(b) Each primary key is also a candidate key, but there may be candidate keys that are not primary keys.	T	☐	☐
(c) Given relations R(A, B) and S(A, B), $R \cap S$ is equal to $R - (R - S)$ and also equal to $S - (S - R)$.	T	☐	☐
(d) Given relations R(A, B) and S(A, B), the natural join $R \bowtie S$ is equal to $R \cap S$.	T	☐	☐
(e) Given relations R(A, B) and S(A, B), $R \cap S$ is equal to $R \times S$.	F	☐	☐
(f) A weak entity set cannot have a primary key.	T	☐	☐
(g) A weak entity must borrow an attribute from another entity set to form a primary key.	T	☐	☐
(h) In the WHERE clause of a SQL query, the condition 'John Doe' <> NULL is evaluated to be true.	F	☐	☐
(i) If a functional dependency $A \rightarrow B$ holds in relation R(A, B, C), then $AC \rightarrow B$ also holds.	T	☐	☐
(j) The BCNF decomposition algorithm does not preserve functional dependencies of the initial relation R.	T	☐	☐

2. [20 pts] Consider the following schema for a social network program.

- Users(<u>userid</u>, email, name, profile)
- Posts(<u>postid</u>, text, userid, num-likes)
- Likes(<u>postid</u>, <u>userid</u>, date-liked)
- Comments(<u>commentid</u>, postid, userid, text)
- Follows(<u>userid</u>, <u>followed-userid</u>, follow-date)

Posts: Each user may create multiple posts. The userid indicates who created the post.
Likes: Each post may get likes. A user can like a post only once.
Comments: comments made for a specific post (postid) by a user (userid)
Follows: a user (userid) may follow another user (followed-userid)

Write a relational algebra for the following queries.

- (a) Query the names of users who have at least one post with 100 or more likes (i.e., num-likes >= 100).

answer: $\Pi_{name}(Users \bowtie \sigma_{num-likes>=100}(Posts))$

- (b) Query the text of posts of users who are followed by a user with name='john_doe'.

answer: $\Pi_{text}(\sigma_{Posts.userid=followed-userid}(Posts \times (\sigma_{name='john\_doe'}(Users) \bowtie Follows)))$

- (c) Query the postid for the posts with no likes and no comments.

answer: $\Pi_{postid}(Posts) - (\Pi_{postid}(Likes) \cup \Pi_{postid}(Comments))$

- (d) Query the postid and userid for the posts with no likes and no comments.

answer: $\Pi_{postid,userid}(Posts \bowtie \Pi_{postid}(Posts) - (\Pi_{postid}(Likes) \cup \Pi_{postid}(Comments)))$

3. [20 pts] Write each of the following queries in SQL for the given relations.

- Users(userid, email, name, profile)
- Posts(postid, text, userid, num-likes)
- Likes(postid, userid, date-liked)
- Comments(commentid, postid, userid, text)
- Follows(userid, followed-userid, follow-date)

a. Retrieve the `userid` of all users who have posted more than 5 posts.

answer (5 pts):

```
SELECT userid
FROM Posts
GROUP BY userid
HAVING COUNT(postid) > 5;

.
```

b. Retrieve the `userid` of all users who have never written a post

answer (5 pts):

```
SELECT u.name
FROM Users u
LEFT JOIN Posts p ON u.userid = p.userid
WHERE p.postid IS NULL;

.
```

(Cont'd) Write each of the following queries in SQL for the given relations.

- Users(userid, email, name, profile)
- Posts(postid, text, userid, num-likes)
- Likes(postid, userid, date-liked)
- Comments(commentid, postid, userid, text)
- Follows(userid, followed-userid, follow-date)

c. For each `userid`, find the number of other users they are following.

answer (5 pts):

```
SELECT u.name, COUNT(f.followed-userid) AS number_of_followed_users
FROM Users u
LEFT JOIN Follows f ON u.userid = f.userid
GROUP BY u.name;
```

.

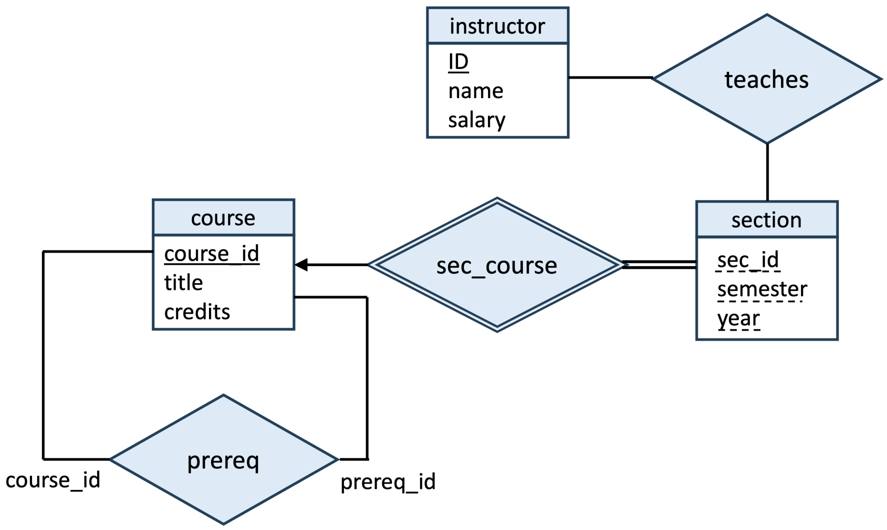
d. Retrieve names of users who have liked more posts than the average user.

answer (5 pts):

```
SELECT u.name
FROM Users u
JOIN Likes l ON u.userid = l.userid
GROUP BY u.name
HAVING COUNT(l.postid) > (
    SELECT AVG(post_count)
    FROM (
        SELECT userid, COUNT(postid) AS post_count
        FROM Likes
        GROUP BY userid
    ) AS average_likes
);
```

.

4. [10 pts] Write a SQL DDL statement to create `teaches` table according to the following ER-diagram.
Note 1. All other tables except `teaches` have been already created. So, only 'teaches' table needs to be created by you.
Note 2. All attributes are of VARCHAR(10) type.



answer:

```
CREATE TABLE teaches (
  ID          varchar(10),
  course_id   varchar(10),
  sec_id      varchar(10),
  semester    varchar(10),
  year        varchar(10),

  primary key (ID, course_id, sec_id, semester, year),

  foreign key (course_id,sec_id, semester, year) references
       section(course_id,sec_id, semester, year) on delete cascade,
  foreign key (ID) references instructor(ID) on delete cascade
};
```

5. [10 pts] Consider the following relations and functional dependencies (FDs) below.

(a) Decompose the following relation into BCNF. Show your steps, and show the keys in the decomposed relations.

- R(A, B, C, D, E)
- FD1: E $\rightarrow$ C
- FD2: BD $\rightarrow$ E

answer (5pts):

```
Step1: ABCDE is decomposed into EC and EABD
Step 2: EABD is decomposed into BDE and BDA

Decomposed relations: EC, BDE and BDA.
```

.

.

(b) Decompose the following relation into 3NF. Show your steps, and show the keys in the decomposed relations.

- R (A, B, C, D, E)
- A $\rightarrow$ BC
- CD $\rightarrow$ E
- B $\rightarrow$ D
- E $\rightarrow$ A

answer (5pts):

A is a candidate key (A -> BC -> D -> E)
E is a candidate key (E -> A -> BC -> D)
CD is a candidate key (CD -> E -> A -> BC)

A->BC: A is a super key. So, A->BC does not violate 3NF.

CD->E: CD is a super key. So, CD->E does not violate 3NF.

B->D: B is not a super key, but D is in a candidate key.
So, it does not violate 3NF.

E->A: E is a candidate key. So it does not violate 3NF.

No decomposition is necessary.

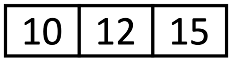
.

6. [10 pts] Suppose you insert 12, 10, 15, 4, 1, 17, 3, 13, 8, and 9 into an initially empty B tree with degree 4 (i.e., the maximum number of child nodes is 4). Draw a tree structure for each step by adding specified keys. To maintain consistency in answers, you must follow the following rules:

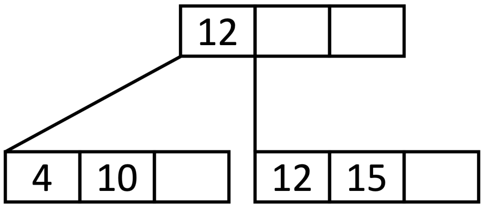
   - You should not use sibling redistribution.
   - When splitting an internal node, keep three children in the left node and two children in the right.

answer (5pts):

(1) Initial State after inserting 12, 10, and 15



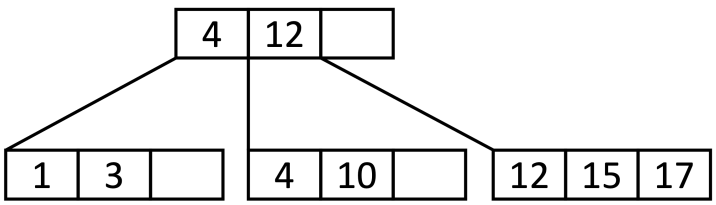
(2) After the 1st split



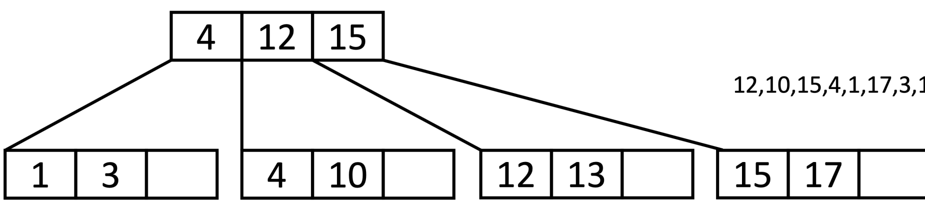
12,10,15,4

(3) After the 2nd split

12,10,15,4,1,17,3

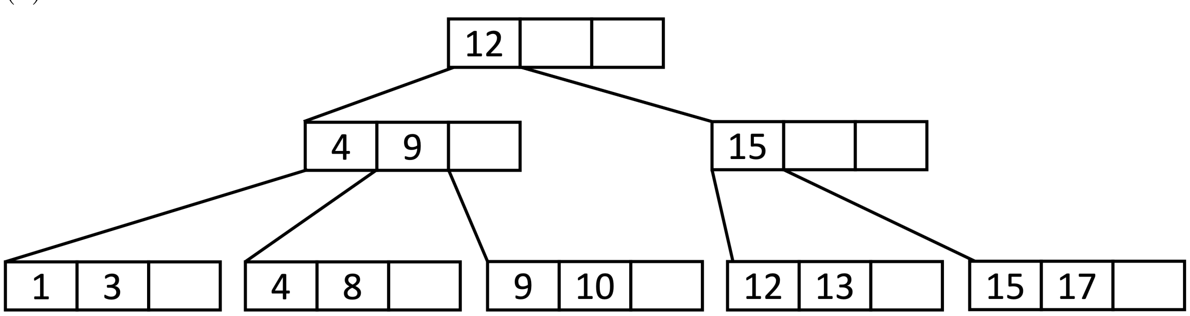


(4) After the 3rd split

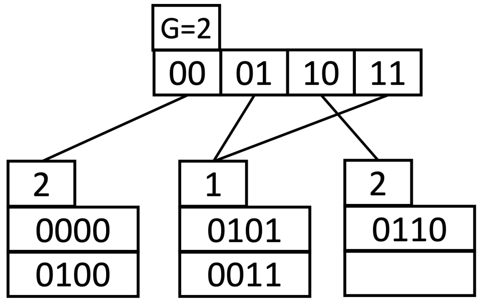


12,10,15,4,1,17,3,13

(5) Final state

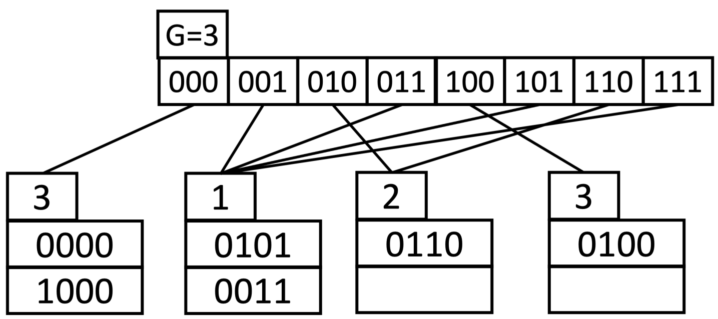


7. [10 pts] Suppose we have an extendable hash table shown in the following figure. Each bucket can hold up to 2 records, and the least significant bits are used along with $\%2^G$ as the hash key.

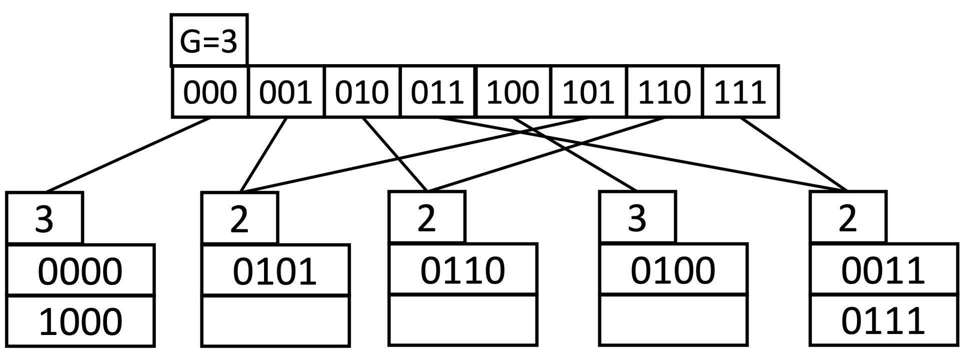


Key	Binary No.
8	1000
7	0111

(1) How will the extendable hash table change after inserting 8?
answer:



(2) How will the extendable hash table change if we insert 7 into the hash table that changed in the previous question?
answer:



8. [5 pts] Suppose you have a computer with tiny 4 MB DRAM. (Yes, that's MB, not GB.) Your job is to find common words in two input files, each of which is 16 GB in size. How many disk blocks transfers are required approximately? **Justify your answer.** We do not distinguish between read and write operations, and each is counted as a single transfer.

answer:
If we use External Sort-Merge-based Algorithm, in total, 3 passes are required for each file.
1st pass:
2 (read/write) x 16GB/4KB = 2 x 4096 = 8,192 block transfer
2nd pass:
After the 1st pass, we have 4096 runs. With 4MB DRAM (1024 x 4KB), we can merge-sort only 1023 runs.
2 (read/write) x 16GB/4KB = 2 x 4096 = 8,192 block transfer
3rd pass:
After the 2nd pass, we have only 4 runs.
1 read x 16GB/4KB = 4,096 block transfer

Therefore, 40,960 ((8192+8192+4096)x2) block transfers are required to compare the two sorted runs.

9. [10 pts] Suppose there is a query that joins `student` table and `department` table. `student` table has 1,000 records and `department` table has 100 records. Each disk page holds maximum 10 student records or 10 department records.

(1) What is the expected number of block transfers if we employ Nested-Loop Join?
answer:
- student table has 100 blocks.
- department table has 10 blocks.
Assuming the worst case:
If student is used as the outer relation, $1000 \times 10 + 100$, i.e., 10100 block transfers are required.
If department is used as the outer relation, $100 \times 100 + 10$, i.e., 10010 block transfers are required.
If the smaller table (department) fits entirely in memory, $100 + 10 = 110$ block transfers are required.
(2) What is the expected number of block transfers if we employ Block Nested-Loop Join?
answer: If student is used as the outer relation, $100 \times 10 + 100$, i.e., 1100 block transfers are required.
If department is used as the outer relation, $10 \times 100 + 10$, i.e., 1010 block transfers are required.
If the smaller table (department) fits entirely in memory, $100 + 10 = 110$ block transfers are required.