## 1. Single-choice Questions (24 points, 3 points*8)

**1.1.** The table course(id, credit, pre_course_id) is defined by the following SQL statement:

```
create table course(
    id char(2) primary key,
    credit int,
    pre_course_id char(2),
    foreign key(pre_course_id) references course(id)
    on delete cascade
    on update cascade
);
```

There is an instance of the course table below.

| id | credit | pre_course_id |
|---|---|---|
| 11 | 4 | |
| 22 | 2 | 11 |
| 33 | 3 | |
| 44 | 1 | 22 |
| 55 | 4 | 22 |
| 66 | 3 | 33 |

Executing the following SQL statements one by one:

1) insert into course values ('77', 4, '22');
2) delete from course where id = '11';
3) update course set id = '11' and credit = 2 where id = '33';
4) select count(*) from course where credit > 2;

What is the result of the statement 4) above?

(A) 1 (B) 2 (C) 3 (D) 4

**1.2.** Consider the table course (<u>id</u>, credit, pre_course_id) and the following query:

**select c1.id, c2.id**

**from course as c1, course as c2**

**where c1.pre_course_id = c2.id and c1.credit=4 and c2.credit>2**

Which algebra expression is ***not equivalent*** to above query?

(A) $\Pi_{c1.id,c2.id}\left(\sigma_{c1.credit=4\,\Lambda\,c1.credit>2}\left(\rho_{c1}(course)\bowtie\rho_{c2}(course)\right)\right)$

(B) $\Pi_{c1.id,c2.id}\left(\sigma_{c1.pre\_course\_id=c2.id\,\Lambda\,c1.credit=4\,\Lambda\,c1.credit>2}\left(\rho_{c1}(course)\times\rho_{c2}(course)\right)\right)$

(C) $\Pi_{c1.id,c2.id}\left(\sigma_{c1.pre\_course\_id=c2.id}\left(\sigma_{c1.credit=4}(\rho_{c1}(course))\times\right.\right.$

$\left.\left.\sigma_{c2.credit>2}(\rho_{c2}(course)\right)\right)$

(D) $\Pi_{c1.id,c2.id}\left(\sigma_{c1.pre\_course\_id=c2.id}\left((\sigma_{c1.credit=4}(\rho_{c1}(course)))\times\right.\right.$

$\left.\Pi_{id,pre\_course\_id}(\sigma_{c1.credit>2}(\rho_{c2}(course))))\right)$

**1.3.** For relation schema R (A, B, C, D, E, F) with functional dependencies set F = {A->B, B->C, C->B, D->E, D->F}. Answer the following questions:

Which of the following ones is the correct **candidate key**?

(A) AC

(B) AD

(C) AF

(D) AE

**1.4.** The function dependency set $F = \{A \rightarrow B, A \rightarrow D,\ BC\ \rightarrow E,\ AC\ \rightarrow D\}$.

What is the **canonical cover** of *F*?

(A) $F = \{A \rightarrow BD, C \rightarrow E\}$

(B) $F = \{A \rightarrow BD, AC \rightarrow DE\}$

(C) $F = \{A \rightarrow B, ABC \rightarrow DE\}$

(D) $F = \{A \rightarrow BD, BC \rightarrow E\}$

**1.5.** Suppose there is a relation *r* with 20000 records on which a clustering B$^+$-tree index is constructed on a non-candidate key. Each B$^+$-tree node has a maximum size of 2048 bytes. Each search key occupies 32 bytes, and each pointer occupies 8 bytes. **A selection operation** is performed by scanning the B$^+$-tree index, where the selection condition specifies an equality comparison on the search key. 5 blocks contain records that match the specified search key. In the worst case, how many seeks and block transfers are required for the selection operation?

(A) 4 seeks and 8 block transfers

(B) 4 seeks and 9 block transfers

(C) 5 seeks and 8 block transfers

(D) 5 seeks and 9 block transfers

**1.6.** There are two relations ***r*** and ***s*** to perform **hash join operation**. The relation ***r*** has 20,000 records, and each block can store 50 records of ***r***. The relation ***s*** has 5,000 records, and each block can store 20 records of ***s***. Assume ***s*** is the build input. There is no recursive partitioning. A hash function is used to partition ***r*** and ***s*** into 10 partitions, respectively. Here, two buffer pages are available for each partition during the hash partitioning phase. Please estimate the seeks and blocks for the hash join operation?

(A) 1950 transfers, 650 seeks

(B) 1950 transfers, 1320 seeks

(C) 3300 transfers, 650 seeks

(D) 3300 transfers, 1320 seeks

**1.7.** Suppose that a table with 30000 records is stored in a file, where each file block holds 200 records. A selection operation is performed, where the selection condition specifies an equality comparison on a candidate key. What is the estimated size of the result?

(A) 1 (B) 150 (C) 200 (D) 30000

**1.8.** The following table shows some lock requests made by three transactions, T1, T2, and T3. Lock-S and Lock-X stand for “shared lock” and “exclusive lock”, respectively. At this time, no granted lock is released.

| T1 | T2 | T3 |
|---|---|---|
| Lock-X(A) | | |
| | Lock-S(B) | |
| Lock-X(B) | | |
| | | Lock-X(C) |
| | Lock-S(C) | |
| | | Lock-X(B) |

Which of the following statements is correct?

(A) There is no deadlock.

(B) There is deadlock, because T1 is waiting for T3, and T3 is waiting for T1.

(C) There is deadlock, because T2 is waiting for T3, and T3 is waiting for T2.

(D) There is deadlock, because T1 is waiting for T2, T2 is waiting for T3, and T3 is waiting for T1.

## 2. True or False Questions (12 points, 2 points *6)

**2.1.** It is always possible to losslessly decompose a relation into relations in Third Normal Form and the dependence is preserved. (True or False)

**2.2.** Column storage performs better than raw storage in transaction scenarios where data needs to be updated frequently. (True or False)

**2.3.** When creating a secondary index, if we want to reduce space overhead, we can use a sparse index with one index entry per data block. (True or False)

**2.4.** Index scan algorithms always have a lower time complexity than full table scans. (True or False)

**2.5.** Performing the projection as early as possible can reduce the size of the temporary relation that are generated by joining two relations. (True or False)

**2.6.** Schedules, which are impossible under two-phase locking, can be possible under the tree protocol. (True or False)

## 3. SQL query (16 points)

Consider the following relational schema in the social network:

> **user(<u>user_id</u>, name, gender, age, group_name)**
> **followship(<u>user_id</u>, <u>follower_id</u>)**

The underlined attributes are primary keys, and foreign keys are listed as follows:

The “user_id” and “follower_id” of followship both references the “user_id” in the user table

Write **SQL statements** to answer the following queries.

(1) Write the SQL statement to find the groups (without redundancy) that have male users.

(2) Find the people that are the followers of the user with id "1001" and is older than 18. Output their information including user_id and name.

(3) Output the group(s) that have the most female users. Note that, if there are multiple groups satisfying the condition, output all of them.

(4) In the "game" group, identify users whose number of followers is greater than the average number of followers of users in the "game" group. Output their information including user_id, name, and the number of followers

## 4. Database design (16 points)

This year, the tourism industry is recovering. One popular way to plan trips is to use online travel websites like Tuniu and Xiecheng. To do this, the database of travel websites needs to keep track of information about travel agencies, travel plans, and users.

- The database stores basic information about each **user**, such as user's ID, password, name, and telephone number.
- For each **travel agency** that wants to sell the travel plans, it must upload its name, the legal person's ID number (法人身份证号), the bank card number of the corporate account (对公账户银行卡号), and its location. The travel database would also allocate each agency an ID for identification.
- Each travel agency can issue different **travel plans** and specify the information of the title, the total hour, the description of the travel, and the price.
- Each user can **order** the required travel plan according to his preference. The order information, such as the order ID, the user ID, the travel plan ID, the amount purchased, the total price, and the order time.
- After traveling, the user can **rate** the travel agency, and the travel database should record these rating scores.

(1) Please draw the E-R diagram of the travel database.

(2) Transform the E-R diagram into the relational schemas and specify the primary keys and foreign keys of these relations

## 5. Concurrency Control (8 points)

Consider following schedule ***S*** with five transactions T1, T2, T3, T4, and T5:

***S: r1(A) w2(A) r2(B) w3(B) w3(C) r1(C) w4(B) r4(D) w5(D) r3(D)***

where: ***ri(X)*** means transaction ***Ti*** read data ***X***.

***wi(X)*** means transaction ***Ti*** write data ***X***.

(1) Draw the precedence graph of ***S***.

(2) Explain whether ***S*** is serializable.

(3) Explain whether ***S*** could be generated by the two–phase locking protocol.

## 6. ARIES Recovery. (12 points)

Suppose a DBMS uses the ARIES method for crash recovery. You are given part of the log file at the time of a system crash, where each log record may contain the transaction ID, the page ID and slot number, the original value, and the updated value.

| |
|---|
| 8000: <T1, 6421.1, 10, 20> |
| 8001: <T2 start> |
| 8002: <T3 start> |
| 8003: <T2, 3462.1, 30, 40> |
| 8004: <T3, 7923.1, 50, 60> |
| 8005: <T2, 3462.1, 40, 70> |
| 8006: <T1, 6421.1, 20, 80> |
| 8007: <T3 commit> |
| 8008: checkpoint |

Active Transaction Table

| TXN | LastLSN |
|---|---|
| T1 | 8006 |
| T2 | 8005 |

DirtyPageTable

| PageID | PageLSN | RecLSN |
|---|---|---|
| 3462 | 8005 | 8003 |
| 6421 | 8006 | 8006 |
| 7923 | 8004 | 8004 |

| |
|---|
| 8009: <T4 start> |
| 8010: <T4, 5235.1, 60, 90> |
| 8011: <T1 commit> |
| **End of log at crash** |

Suppose that the PageLSN of each page on disk is less than 8000. Given this information, answer the following questions:

(1) During the analysis pass, what record is added to the DirtyPageTable?

(2) Which log record is the first to be redone in the redo pass?

(3) What log records should be appended to the log file during recovery? For CLR (compensation log records), you should provide the transaction ID, the page ID and slot number, and the original value (i.e., the value to be recovered).

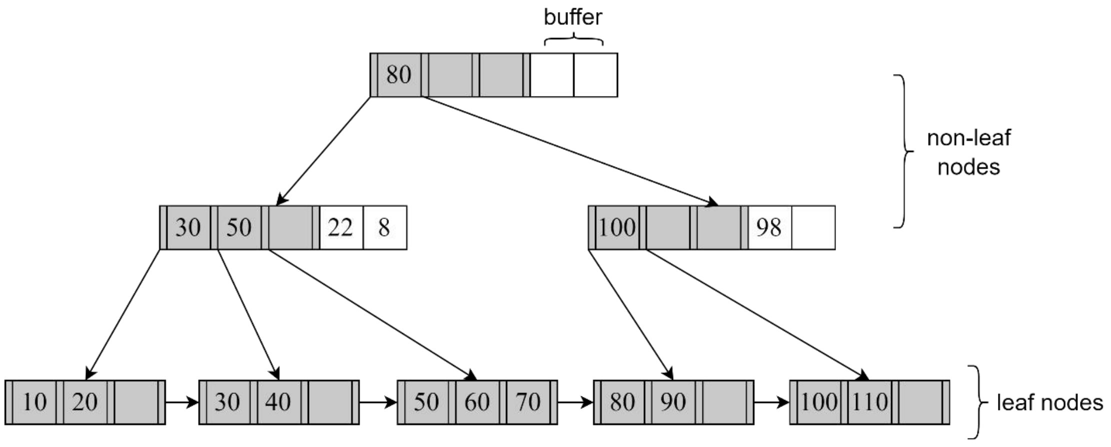
(4) After recovery, what are the values of locations 3462.1 and 6421.1, respectively?

## 7. Buffer Tree. (12 points)

A buffer tree is an index structure that performs updates lazily to save I/O operations. In this question, you are given a simplified version of buffer tree with the following properties:

i. Each non-leaf node consists of a B$^+$-tree node with a max fanout of 4, and a buffer that can hold up to 2 items (i.e., index entries). Meanwhile, each leaf node is a regular B$^+$-tree leaf node that can hold up to 3 items.
ii. When an item is inserted, it is first inserted into the buffer of the root node. (We assume that the root node is not a leaf node.)
iii. If a node's buffer is overfull (i.e., contains more than 2 index entries), the following procedure is performed:
    a. If the node is an internal node, the items are pushed down to the appropriate buffers of child nodes one level below. After all the items are pushed down, if any of the children has an overfull buffer, apply this procedure recursively.
    b. If the node has leaf nodes as children, move the smallest item from the buffer to the appropriate leaf node. If the leaf is overfull, split it in the usual B$^+$-tree manner. Then repeat this step until no item is left in the buffer.

Below is the content of the buffer tree.



Answer the questions below.

(1) For a range query from 30 to 50 (both inclusive), how many index nodes are accessed? Explain the procedure.

(2) Provide a diagram of the buffer tree after inserting 38, 92, and 28, in that order, into the buffer tree.

(3) Assume that each buffer tree node (with the optional buffer) is stored in a disk block. The root node is cached in memory, so it is excluded from the following calculation. How many blocks are modified for each insert operation in question (2)? If a block is created, count it as a modified block.