University of Glasgow

Friday 1 May 2020, 14:00 BST
(24 hour open online assessment – Indicative duration 1.5 hours)

DEGREES of MSci, MEng, BEng, BSc, MA and MA (Social Sciences)

Database Systems H
COMPSCI 4013

(Answer All 4 Questions)

This examination paper is worth a total of 60 marks

Part A: Relational Modelling and SQL [Total: 20 Marks]

Question 1.

(a). Can the foreign key of a relation be `NULL`? Explain briefly your answer. [5]

(b). Which is the expected result of the following query? Explain briefly your answer [5]

```
SELECT * FROM EMPLOYEE WHERE SSN NOT IN (NULL, 1, 2)
```

(c). Consider the following relational schema:

Researcher(<u>ResearcherID</u>, Name, Surname, Institution)

Activity(<u>PID, RID</u>)

Publication(<u>PublicationID</u>, PublicationYear, Title, Type)

The attribute RID is a *foreign key* in Activity referencing to ResearcherID in Researcher relation, and the attribute PID is *a foreign key* in Activity referencing to PublicationID in Publication relation. The primary key is underlined in each relation.

For each researcher who has written more than 140 publications, show how many of these publications have been published since 2020. [10]

Part B: File Organization [20 Marks]

Question 2. Assume the relation EMPLOYEE(SSN, Name, Address, DNO) which is stored in a file on a disk. We need 100 bytes for the integer Primary Key attribute SSN (Social Security Number), 100 bytes for the Name attribute, 50 bytes for the Address attribute, and 6 bytes for the department number (DNO) attribute. Consider that the relation has only $r = 7$ tuples and the size of the file block is 512 bytes.

(a). Given that the data management system adopts fixed-length records, such that each file record corresponds to each tuple of the relation, which will be the blocking factor (bfr) of the file block to accommodate the relation EMPLOYEE? [1]

(b). We adopt external hashing for storing the relation EMPLOYEE using M = 4 buckets and the hash function $y = h(\text{SSN}) = \text{SSN mod M}$, i.e., we hash the key SSN attribute. We assume that the size of each bucket is equal to the size of a block, i.e., 512 bytes, and that *all* buckets are equiprobable to be selected given any random selection query over the SSN attribute.

(i). Design the structure of the M = 4 buckets of the corresponding hash file (which tuples belong to which bucket) that accommodates $r = 7$ tuples with SSN values: **0, 2, 3, 4, 6, 8, 9**. If a bucket is *full*, then you can use overflow buckets connected through *chain pointers*. [2]

(ii). Calculate the *expected* number of block accesses (I/O blocks read/write) for a random SQL selection query using the hashing structure:

```
SELECT * FROM EMPLOYEE WHERE DNO = x,
```

for any random *x* value, in the *best-case* scenario, *worst-case* scenario, and *average-case* scenario. [6]

Note: In the best-case scenario, the tuple is found in the *main* bucket, while in the worst-case scenario the tuple is found in the *last* overflown bucket of the block chain, if exists.

(c). Calculate the expected number of block accesses for the selection query in **Question 2(b).(ii)** assuming that the EMPLOYEE relation is stored in a **Sequential File** ordered by SSN and in a **Heap File**, in the *best-case* scenario and *worst-case* scenario. [3]

(d). Assume that we issue many selection queries over the relation EMPLOYEE such that p% of those queries involve the SSN attribute and (1-p)% do not; p value is in [0,100].

(i). Provide a decision making rule based on the p% value such that we should store the relation in a sequential file sorted by SSN instead of hashing the file with respect to SSN. [4]

(ii). Provide a decision making rule based on the p% value such that we should store the relation in a sequential file sorted by SSN instead of storing the relation in a heap file. [4]

Note: Base your reasoning on the *worst-case expected* cost for executing any random query over the relation EMPLOYEE (in terms of block accesses).

Part C: Query Processing and Optimization [20 Marks]

Question 3. Consider the relation EMPLOYEE(SSN, Salary, DNO), where SSN is the social security number and DNO is the department number, and the selection query:

```
SELECT * FROM EMPLOYEE WHERE
(10000 <= Salary) AND (Salary <= 40000) AND (DNO = 1 OR DNO = 3)
```

Consider also the following information of the database system:

- **Clustering Index** on the Salary non-key ordering attribute with $x_{\text{Salary}} = 3$ levels.
- **B+ Tree Secondary Index** on the DNO non-key non-ordering attribute with $x_{\text{DNO}} = 2$ levels. We need only 1 block with data block-pointers per DNO value.
- The number of the distinct values of DNO is 5.
- The maximum and minimum Salary values are 100000 and 10000, respectively.
- The relation EMPLOYEE has $r = 10000$ tuples, the corresponding file has $b = 5000$ blocks, the blocking factor is $f = 2$ records/block.
- The available memory in the database system is 2000 blocks.

For convenience: 9/25 = 0.36, 3/25 = 0.12

(a). Estimate the selection cardinality of the selection query, i.e., the number of tuples that satisfy the condition in the WHERE clause. [3]

(b). Propose **two** *query processing plans* for implementing the selection query using the provided access paths over the Salary and DNO attributes and select the *best* in terms of the number of block accesses. [7]

Question 4. Consider the relation EMPLOYEE(SSN, Salary) of $r = 5600$ employees (tuples), where each attribute has size 7 bytes. There are $n = \text{NDV(Salary)} = 280$ distinct salary values in the relation, the size of the block is B = 64 bytes and the size of any block pointer is P = 7 bytes. The Salary values are *uniformly* distributed (all values are equiprobable). We have *only* built a B+ Tree (Secondary Index) of level t over the non-ordering, non-key Salary attribute, such that each leaf node fits in *one* block.

(a). Calculate the leaf node order q of the B+ Tree; for each leaf node, this corresponds to q-1 index values, q-1 block pointers, and *one* pointer to the sibling leaf node. How many leaf nodes does the B+ Tree have? [3]

(b). Consider the following selection query:

```
SQL1: SELECT * FROM EMPLOYEE WHERE Salary = 50000
```

Which is the expected cost (in terms of block accesses) for executing the query SQL1 using the B+ Tree on the Salary attribute? Express this cost as a function of the B+ Tree level t. **Note:** it is not required to calculate the level t of the B+ Tree. [6]

(c). Consider the following selection query:

```
SQL2: SELECT * FROM EMPLOYEE WHERE Salary = 50000 OR Salary = 60000
```

Express the expected cost (in terms of block accesses) for executing the query SQL2 as a function of the B+ Tree level t. **Note:** it is not required to calculate the level t of the B+ Tree. [1]