**Friday 10 May 2019**
**9.30 am – 11.30 am**
**Duration: 2 Hours**

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

# Database Systems H

**(Answer All Questions)**

**This examination paper is worth a total of 60 marks**

**The use of a calculator is not permitted in this examination**

**INSTRUCTIONS TO INVIGILATORS**

**Please collect all exam question papers and exam answer scripts and retain for school to collect. Candidates must not remove exam question papers.**

**Note: The Exam Duration is Two (2) HOURS**

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

**1.** **(a)** Can the foreign key of a relation be NULL? Explain briefly your answer.

[2]

**(b)** Consider the following SQL CREATE TABLE statements:

| | |
|---|---|
| CREATE TABLE Runner (<br>RunnerID INT NOT NULL,<br>RunnerName VARCHAR(50),<br>PRIMARY KEY(ID)); | CREATE TABLE Race (<br>RaceID INT NOT NULL,<br>RaceEvent VARCHAR(10),<br>WinnerID INT,<br>PRIMARY KEY(RaceID),<br>FOREIGN KEY(WinnerID) REFERENCES Runner(RunnerID)); |

We issue the SQL1 query:

**SQL1: SELECT * FROM Runner,**

and obtain the results in Table 1. Then, we issue the SQL2 query:

**SQL2: SELECT * FROM Race,**

and obtain the results in Table 2.

Table 1

| RunnerID | RunnerName |
|---|---|
| **1** | Chris |
| **2** | John |
| **3** | Ian |
| **4** | Alice |
| **5** | Jane |

Table 2

| RaceID | RaceEvent | WinnerID |
|---|---|---|
| **1** | 100 meter dash | 2 |
| **2** | 500 meter dash | 3 |
| **3** | Triathlon | NULL |
| **4** | Triathlon | NULL |

**(i).** Based on the results in Table 1 and Table 2, which will be the results if we issue the **SQL3**?

Explain briefly you answer.

[5]

```
SQL3: SELECT * FROM Runner
      WHERE RunnerID NOT IN (SELECT WinnerID FROM Race);
```

**(ii).** If you raise any issues with **SQL3** in 1.b.(i), provide a modified version of the **SQL3**, which fixes the named issues. Explain briefly your answer.

[3]

**2.** Consider the following relational schema:

**Author(AuthorID, Name)**

**Authoring(ARTID, AID)**

**Article(ArticleID, PublicationYear, Title)**

where the attribute AID is a *foreign key* in Authoring referencing to AuthorID in Author relation, and the attribute ARTID is *a foreign key* in Authoring referencing to ArticleID in Article relation. The primary key is underlined in each relation.

**(a)** Given that an author can have written one or more articles, write a SQL query that **shows the number of co-authors with the author 'Chris'**, i.e., the number of authors who have written at least one article together with Chris.

[5]

**(b)** For each author who has written more than 50 articles, show how many of these articles have been published since 2016.

[5]

## Part B: File Organization and Indexing Methods [20 Marks]

**3.** Assume the relation **EMPLOYEE**(<u>ID</u>, Name, Address, DNO) which is stored in a file on a disk. We need 100 bytes for the integer Primary Key attribute ID, 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 database 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 = 3 buckets and the hash function $\boldsymbol{y = h(\mathrm{DNO}) = \mathrm{DNO} \bmod \mathrm{M}}$, i.e., we hash the DNO 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 DNO attribute.

**(i)** Design the structure of the M = 3 buckets of the corresponding hash file (which tuples belong to which bucket) that accommodates $r = 7$ tuples with DNO 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:

```
SQL4: SELECT * FROM EMPLOYEE WHERE DNO = x,
```

for any random $x$ value, in the *best-case* scenario and *worst-case* scenario.

[3]

**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.

**(iii)** Calculate the expected number of block access for the SQL4 query in **Question 3(b).(ii)** assuming that the EMPLOYEE relation is stored in a **Sequential File** ordered by DNO and in a **Heap File**, in the *best-case* scenario and *worst-case* scenario.

[2]

**(iv)** Given that the EMPLOYEE relation has *only* the tuples whose DNO values are provided in **Question 3(b).(i)**, calculate the expected number of block accesses for the range query:

```
SQL5: SELECT * FROM EMPLOYEE
      WHERE DNO >= 3 AND DNO <= 8
```

using: your Hash File Stricture in **Question 3(b).(i)**, a Sequential File ordered by DNO, and a Heap File, in the *worst-case* scenario.

[3]

**4.** Consider a clustering index over the **ordering non-key** DNO attribute (department number) of the relation **EMPLOYEE**(<u>SSN</u>, Name, DNO). The size of the DNO attribute is 5 bytes, while the total tuple size of the relation is R = 100 bytes. The relation has r = 1000 tuples, the block size is B = 256 bytes, and a pointer has size P = 5 bytes. There are n = 4 departments, where the employees are *uniformly* distributed over the departments.

A data analyst investigates the expected cost in terms of block accesses of the following SQL6 query given **any random** DNO value *x* and decides whether to use the clustering index or a serial scan of the file for that query.

**SQL6: SELECT * FROM EMPLOYEE WHERE DNO =** *x*

**(a)** Calculate the blocking factor of the data file (bfr) and the clustering index file (ibfr).

[1]

**(b)** Calculate the number of blocks of the data file and the number of blocks of the clustering index file.

[1]

**(c)** Which is the additional storage due to the clustering index?

[1]

**(d)** Given the SQL6 query, which is the expected number of block accesses using the clustering index?

[1]

**(e)** Given the SQL6 query, which is the expected number of block accesses using serial file scan? Based on this information, which is the final decision for the data-analyst?

[5]

## Part C: Query Processing and Optimization [20 Marks]

**5.** Consider the relation EMPLOYEE(SSN, Salary, DNO), where SSN is the social security number and DNO is the department number, and the selection query SQL7:

```
SQL7: SELECT * FROM EMPLOYEE
      WHERE Salary >= 35000 AND DNO = 5
```

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 125.
- The maximum and minimum Salary values are 50000 and 5000, respectively.
- The relation EMPLOYEE has $r = 10000$ tuples, the corresponding file has $b = 2000$ blocks, the blocking factor is $f = 5$ records/block.
- The available memory in the database system is 100 blocks.

**For convenience: 1/125 = 0.008; 0.008/3 = 0.0027**

**(a)** Estimate the selection cardinality of the SQL7 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 SQL7 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]

**6.** Consider two relations **EMPLOYEE** E and **DEPENDENT** P such that:

- relation E has $n_E$ = 100 blocks and $r_E$ = 1000 records,
- relation P has $n_P$ = 50 blocks and $r_P$ = 50 records.

The SSN (Social Security Number) is the Primary Key of the relation Employee (E). The non-unique attribute E_SSN is the Foreign Key in relation Dependent (P) referencing to relation E on the SSN attribute. The number of distinct values of the E_SSN is NDV(E_SSN) = 10. The blocking factor for the join-results block is bfrRS = 10 records per block. Assume that we join the two relations E and P with respect to the primary-foreign keys SSN and E_SSN, as shown in the following SQL-JOIN.

```
SQL-JOIN: SELECT * FROM EMPLOYEE E, DEPENDENT P
          WHERE E.SSN = P.E_SSN
```

**(a)** Estimate the *join cardinality* of the SQL-JOIN, i.e., the number of matching tuples.

[2]

**(b)** Estimate the *total* expected cost in I/O block accesses based on the naïve join strategy, i.e., the strategy producing the Cartesian product.

[2]

**(c)** Assume that there is a **Clustering Index** over the non-unique ordering attribute E_SSN of the Dependent (P) relation with level $\boldsymbol{x_P = 2}$, the number of the distinct values of the E_SSN attribute is **NDV(E_SSN) = 10**. Moreover, assume a **Primary Index** over the Employee (E) relation on the primary key SSN with level $\boldsymbol{x_E = 2}$. Propose **two index-based nested-loop join strategies** and select the *best* in terms of block accesses.

[6]