UNIVERSITY OF MORATUWA

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

BSc Engineering Honours Degree
Semester 4 Examination: 2018

CS3042: DATABASE SYSTEMS

Time allowed: 2 Hours December 2018

ADDITIONAL MATERIAL: *None*

INSTRUCTIONS TO CANDIDATES:

1. This paper consists of **4** questions in **7** pages.
2. Answer all **4** questions.
3. Start answering each question on a new page.
4. The maximum attainable mark for each question is given in brackets.
5. This examination accounts for 50% of the module assessment.
6. This is a closed book examination.

   NB: It is an offence to be in possession of unauthorised material during the examination.
7. Only calculators approved and labelled by the Faculty of Engineering are permitted.
8. Assume reasonable values for any data not given in or with the examination paper. Clearly state such assumptions made on the script.
9. In case of any doubt as to the interpretation of the wording of a question, make suitable assumptions and clearly state them on the script.
10. This paper should be answered only in English.

Question 1

(a) Higher Education Ministry is planning to use an online system to disburse Mahapola scholarships. In point form; briefly state the three (03) most significant advantages that can be gained by using a DBMS for record keeping for this application in comparison to using a traditional file system. [6]

(b) Using an example state and show why a parity bit is used in some RAID configurations. [3]

(c) Why do we have Foreign Key Constraints in a databases? [3]

availability
low cost
semi struc. data

(d) State three (3) reasons for the emergence of NoSQL databases [3]

(e) Why do we store statistics about the data tables in a Database ? [3]

(f) Can we have dense primary indexes in databases? Briefly explain your answer. [3]

(g) Using an example briefly explain a problem a database may face due to a transactions not maintaining the ACID property "Isolation." You do not need to state the property. [4]

Question 2

(a) Consider the relation r(L,M,N) shown below to answer the questions.

L	M	N
21	24	23
23	24	26
23	24	21
27	23	28
29	21	20

Indicate if the following Functional Dependencies are satisfied by the relation.

I. $L \rightarrow N$
II. $LM \rightarrow N$
III. $NM \rightarrow L$ ✓

[2x3]

(b) Given that $S \rightarrow P$, $Q \rightarrow S$, $QR \rightarrow ST$, $P \rightarrow QRS$ are satisfied in r(P,Q,R,S,T,U), Prove that **PU** is a superkey. [10]

(c) BCNF is a higher normal from than 3NF. But in practice, in some cases, why do we leave the database tables at 3NF? Briefly justify your answer. [4]

(d) Given that $L \rightarrow M$, and $L \rightarrow N$ functional dependencies are satisfied in relation r=(L,M,N), can we find a lossless, dependency preserving decomposition? Justify your answer. [5]

refundable

Question 3

Following queries are used to create a schema to store data for research grants from the National Science Foundation.

```
create table organization (
      id int primary key,
      name varchar(100),
      streetaddr varchar(200),
      city varchar(100),
      state char(2),
      zip char(5),
      phone char(10)
);
create table researchers (
      id int primary key,
      name varchar(100),
      org int references orgs(id)
);
create table programs (
      id int primary key,
      name varchar(200),
      directorate char(3) /* the top-level part of NSF that
runs this program */
);
create table managers (
      id int primary key,
      name varchar(100)
);
create table grants (
      id int primary key,
      title varchar(300),
      amount float,
      org int references orgs(id),
      pi int references researchers(id), /*the principal
investigator (PI) on the grant */
      manager int references managers(id),
      started date,
      ended date,
      abstract text
);
create table grant_researchers (
      researcherid int references researchers(id),
      grantid int references grants(id)
);
create table fields (
      id int primary key,
      name varchar(100)
);
create table grant_fields (
      grantid int references grants(id),
      fieldid int references fields(id)
```

```
);
create table grant_programs(
      grantid int references grants(id),
      programid int references programs(id)
);
```

Assume that you have created all the tables using the above SQL statements in a relational database that implements standard SQL and answer the following questions. Assume that the tables are populated with the data needed to answer the following questions.

(a) Write an appropriate SQL statement to find the title, amount and abstract of the grants for which Professor Geoffrey is the principal investigator. [2]

(b) Write an appropriate SQL statement to find the number of times a Stanford researcher has received a grant over $1million but less that $2 million since 1/1/2008. Retrieve the name of the researcher and the number of such grants awarded. [3]

(c) Write an appropriate SQL statement to find the name of the grant with the smallest id. [3]

(d) Write an appropriate SQL statement that finds the organizations that have received at least 10 grants per year. Report the name of the organization, the number of grants and the years they received the grants. (Hint: you can use the expression year(date) to get the year from a date column.) [4]

(e) It turns out that many grants are multidisciplinary (i.e., spans multiple fields). Among the many pairs of fields (e.g., computer science and physics) that have shared grants, find the field name that is granted with maximum number of grants. [2]

(f) SQL supports saving the results of a query as temporary table. This table can be queried just like a normal table, providing similar functionality to nested queries, with the ability to reuse the results of a query. The command to create a temporary table is:

CREATE TEMPORARY TABLE name AS SELECT...

Using temporary tables find the managers who have given the most amount of money to a single organization. For the 5 highest amounts, retrieve the manager name, organization name and total amount given. [6]

(g) Write corresponding relational algebra expressions for all the above queries (a), (b), (c), (d) and (e). [5]

Question 4

You are required to design a database that can be used to build a management information system for an Online Food Chain Company. The company has headquarters in Colombo and has branches all over the country. Branches are characterized by a unique ID and address. Customers can place their orders online (through the web or mobile app). Regular customers can have an account which is identified by a unique ID which does not change over time. Also, customers can log in as a guest. In such cases, customers are given an ID which is valid only for that transaction. Dispatched orders should have a unique scheduler id, delivery person ID etc. The

Foodchain company head office wants to track all the orders happening all over the country and want to provide discounts for regular customers. You can assume that each food item has a unique ID and the description of the food items are already stored in a separate database.

(a) Based on the above description and any other required assumption (assumptions should be clearly indicated) using an ER/EER diagram give a pictorial representation for the design of the database system. [20]

(b) Write down two (2) types of database constraints that are needed to be imposed on the above system and give at one example for each of them. [5]

check primary not null ~~unique~~

If required you may use the following algorithms/axioms/rules.

Algorithm to compute α^+, the closure of α under F

```
result := α;
while (changes to result) do
        for each β → γ in F do
            begin
                if β ⊆ result then result := result ∪ γ
            end
```

- Armstrong's Axioms:
  - if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$ (reflexivity)
  - if $\alpha \rightarrow \beta$, then $\gamma\alpha \rightarrow \gamma\beta$ (augmentation)
  - if $\alpha \rightarrow \beta$ and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ (transitivity)

- Additional rules:
  - If $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow \beta\gamma$ holds (union)
  - If $\alpha \rightarrow \beta\gamma$ holds, then $\alpha \rightarrow \beta$ holds and $\alpha \rightarrow \gamma$ holds (decomposition)
  - If $\alpha \rightarrow \beta$ holds and $\gamma\beta \rightarrow \delta$ holds, then $\alpha\gamma \rightarrow \delta$ holds (pseudotransitivity)

To compute a canonical cover for F:

```
repeat
        Use the union rule to replace any dependencies in F
                α1 → β1 and α1 → β2 with α1 → β1 β2
        Find a functional dependency α → β with an
                extraneous attribute either in α or in β
            /* Note: test for extraneous attributes done using Fc not FY
        If an extraneous attribute is found, delete it from α → β
until F does not change
```

You may use the following SQL Syntax if required to answer the questions

```
CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl_name
    { LIKE old_tbl_name | (LIKE old_tbl_name) }

create_definition:
    col_name column_definition
  | [CONSTRAINT [symbol]] PRIMARY KEY [index_type] (index_col_name,...)
      [index_option] ...
  | {INDEX|KEY} [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] UNIQUE [INDEX|KEY]
      [index_name] [index_type] (index_col_name,...)
      [index_option] ...
  | {FULLTEXT|SPATIAL} [INDEX|KEY] [index_name] (index_col_name,...)
      [index_option] ...
  | [CONSTRAINT [symbol]] FOREIGN KEY
      [index_name] (index_col_name,...) reference_definition
  | CHECK (expr)


column_definition:
    data_type [NOT NULL | NULL] [DEFAULT default_value]
      [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY]
      [COMMENT 'string']
      [COLUMN_FORMAT {FIXED|DYNAMIC|DEFAULT}]
      [STORAGE {DISK|MEMORY|DEFAULT}]
      [reference_definition]


data_type:
    BIT[(length)]
  | TINYINT[(length)] [UNSIGNED] [ZEROFILL]
  | SMALLINT[(length)] [UNSIGNED] [ZEROFILL]
  | MEDIUMINT[(length)] [UNSIGNED] [ZEROFILL]


index_col_name:
    col_name [(length)] [ASC | DESC]


index_type:
    USING {BTREE | HASH}


index_option:
    KEY_BLOCK_SIZE [=] value
  | index_type
  | WITH PARSER parser_name


reference_definition:
    REFERENCES tbl_name (index_col_name,...)
      [MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]
      [ON DELETE reference_option]
      [ON UPDATE reference_option]


reference_option:
    RESTRICT | CASCADE | SET NULL | NO ACTION
```

```
SELECT
    [ALL | DISTINCT | DISTINCTROW ]
      [HIGH_PRIORITY]
      [STRAIGHT_JOIN]
      [SQL_SMALL_RESULT] [SQL_BIG_RESULT] [SQL_BUFFER_RESULT]
      [SQL_CACHE | SQL_NO_CACHE] [SQL_CALC_FOUND_ROWS]
    select_expr [, select_expr ...]
    [FROM table_references
    [WHERE where_condition]
    [GROUP BY {col_name | expr | position}
      [ASC | DESC], ... [WITH ROLLUP]]
    [HAVING where_condition]
    [ORDER BY {col_name | expr | position}
      [ASC | DESC], ...]
    [LIMIT {[offset,] row_count | row_count OFFSET offset}]
    [PROCEDURE procedure_name(argument_list)]
    [INTO OUTFILE 'file_name' export_options
      | INTO DUMPFILE 'file_name'
      | INTO var_name [, var_name]]
    [FOR UPDATE | LOCK IN SHARE MODE]]

table_references:
    table_reference [, table_reference] ...

table_reference:
    table_factor
  | join_table


table_factor:
    tbl_name [[AS] alias] [index_hint]]
  | table_subquery [AS] alias
  | ( table_references )
  | { OJ table_reference LEFT OUTER JOIN table_reference
       ON conditional_expr }


join_table:
    table_reference [INNER | CROSS] JOIN table_factor [join_condition]
  | table_reference STRAIGHT_JOIN table_factor
  | table_reference STRAIGHT_JOIN table_factor ON conditional_expr
  | table_reference {LEFT|RIGHT} [OUTER] JOIN table_reference
join_condition
  | table_reference NATURAL [{LEFT|RIGHT} [OUTER]] JOIN table_factor


join_condition:
    ON conditional_expr
  | USING (column_list)


index_hint:
    USE {INDEX|KEY} [FOR JOIN] (index_list)
  | IGNORE {INDEX|KEY} [FOR JOIN] (index_list)
  | FORCE {INDEX|KEY} [FOR JOIN] (index_list)


index_list:
    index_name [, index_name] ...
```

--- END OF PAPER ---