11/1/2019

09.30 - 11.30am

National Stadium, Irish Athletic Boxing Centre

DUBLIN INSTITUTE OF TECHNOLOGY

DT228 BSc. (Honours) Degree in Computer Science

DT282 BSc. (Honours) Degree in Computer Science (International)

Year 2

WINTER EXAMINATIONS 2018/2019

DATABASES I [CMPU2007]

Ms. DeirdreLawless
Dr. Deirdre Lillis
Mr. Patrick Clarke

FRIDAY 11TH JANUARY 9.30 A.M. – 11.30 A.M.

TWO HOURS

INSTRUCTIONS TO CANDIDATES

Answer **ALL** Questions from **SECTION A**
AND
Answer **ONE** Question from **SECTION B**

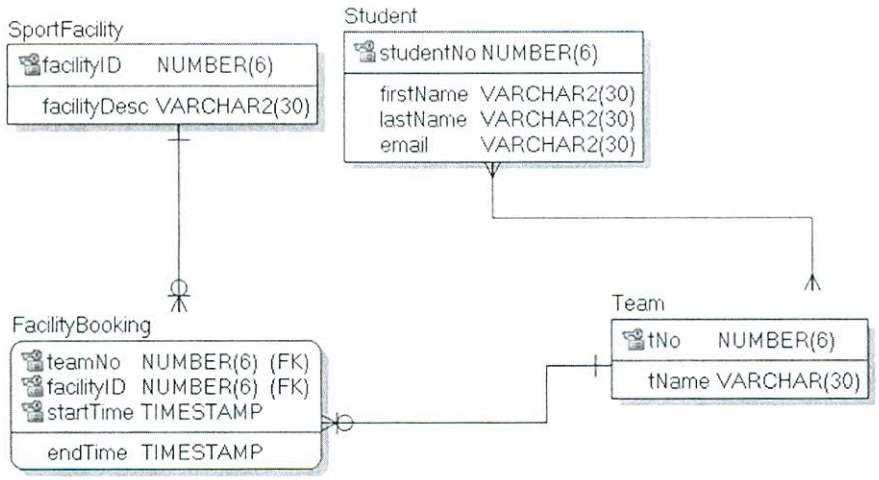
There is a SYNTAX TABLE at the end of the paper to assist you.

SECTION A

SECTION A

1. (a) The Technological University, Dublin wants to create a relational database (SportDB) as part of a new application to manage bookings of a range of sporting facilities that can be used for training by student sport teams. The sports facilities are many and varied e.g. pitches, swimming pools, gym studios, tennis courts, pool tables etc. When a booking is made the start date and time and the end date and time are stored. The university has a number of sports teams that students can join. Each student can be a member of many teams and a team can have many members.

The logical relational model of the database in its current form is shown below:



(i) For each of the following concepts provide a clear *explanation of the concept* and *identify an example* of it from the diagram above:

- Entity
- Attribute
- Compound Primary Key
- Foreign Key

(4 x 3 marks)

(ii) When transforming from a logical to a physical model for SportDB an entity `TeamMember` is introduced. `TeamMember` has as attributes only a compound primary key of studentNo and tNo both of which are foreign keys to the relevant tables.

Clearly explain what *type of entity* `TeamMember` is and *why it is needed* to transform the logical model for SportDB to a physical model.

(6 marks)

Question One continues on the next page

SECTION A

1. (b) Suppose the relational database described in Q1 (a) SportDB has been created on an Oracle server and data inserted into the tables. A number of changes to the database are required.

(i) The university employs a number of coaches. A new table `Coach` needs to be added to the database to store details of coaching staff. The columns required are `coachID`, a unique numeric identified capable of storing values up to 999999; `coachName` to store a coach's name, capable of storing values up to 50 characters which cannot be null; and `coachEmail` to store a coach's email address which can store values up to 30 characters in length, which can have no value but which must end in `@sport.tudublin.ie` if a value is provided.

Write the SQL to `CREATE` the `Coach` table implementing the correct `primary key` and `value constraints` needed.

(15 marks)

(ii) Write the SQL needed to `insert` the following data into the `Coach` table.

coachID	coachName	coachEmail
101	Michel Barnier	mb@sport.tudulin.ie
102	Nigel Farage	(null)
103	Teresa May	tm@sport.tudublin.ie

(5 marks)

(iii) Each team has a single coach. A coach can coach many teams.

Write the SQL to `add a column coachID` to the `Team` table to facilitate this.

(5 marks)

(iv) Write the SQL needed to add a `constraint` to `Team` table to ensure consistency with the `Coach` table. Ensure you provide a name for your constraint.

(5 marks)

(v) Write the SQL needed to *retrieve* for each coach `their name` and `email address`. If an email address has no value then `Unknown` should be output. *Hint: function needed.*

(6 marks)

(c) For each of the following explain what each type of integrity allows you to achieve in a database and provide an example of each from your answers to Q1 (a) or (b).

- Domain integrity
- Referential integrity

(2 x 3 marks)

SECTION A

2. Suppose we have cleared all data from the tables and inserted data as follows:

SportFacility

FACILITYID	FACILITYDESC
101	Swimming Pool
102	Gym Studio 1
103	Gym Studio 2
104	Outdoor Lake

Coach

COACHID	COACHNAME	COACHEMAIL
102	Nigel Farage	(null)
103	Teresa May	tm@sport.tudublin.ie
101	Michel Barnier	mb@sport.tudublin.ie

Team

TNO	TNAME	COACHID
101	Rowing	101
102	Boxing	102
103	Archery	103
104	Sailing	101

Student

STUDENTNO	FIRSTNAME	LASTNAME	EMAIL
1001	Weyand	Sabine	SB@mytudublin.com
1021	Davis	Derek	DD@myStudublin.com
1031	Rabb	Dominic	AR@mytudblin.com
1041	Johnson	Boris	BJ@mytudublin.com
1051	Riso	Stephanie	SR@mytudublin.com

TeamMember

STUDENTNO	TNO
1001	101
1051	101
1031	102
1031	103
1041	103
1041	102
1051	104

FacilityBooking

FACILITYID	TEAMNO	STARTTIME	ENDTIME
101	101	01-MAY-2018 12:05	01-MAY-2018 01:05
102	102	01-MAY-2018 12:05	01-MAY-2018 01:05
101	101	10-MAY-2018 04:05	10-MAY-2018 06:05
102	103	01-MAY-2018 12:05	01-MAY-2018 01:05

(a) *Write the SQL* to retrieve for each sporting facility:

- the `facility identifier`
- the `start date and time`
- the `end date and time` of the booking and
- the `name of the team` the booking is for.

(6 marks)

(b) *Explain* the *type of join* needed in part (a) and *how the SQL works*.

(4 marks)

(c) *Amend the SQL* you wrote for part (a) to *use* `Oracle PL/SQL functions` and *format the output* as follows:

- `Facility description` should be output in `lowercase` in a column named `Sports Facility`;
- `Team name` should be output in `uppercase` in a column called `Is Booked For Team`
- `Start date and time` should be output formatted as DDTH MONTH YYYY HH:MM in a column called `Booked From`.
- `End date and time` should be output formatted as DD/MONTH/ YYYY HH:MM:SS in a column called `Booked To`.

(10 marks)

SECTION B

SECTION B

3. Suppose the data in the database is as given in Q2 (a).

(a) *Write SQL* to calculate for each sports facility that *has been booked* the `total number` bookings for that facility.

You need to:

- *Format your output* to follow this template:

  `Facility <facility description> is booked <no. of bookings> times.`

  NOTE: You do not include the < > in your output. Retrieve facility description, and calculate the no. of bookings.

- *Output* <u>*one*</u> row for each facility *that has a booking.*
- *Sort* the output in `ascending order` of `number of bookings`;
- *Explain* how the SQL works.

Hint: Requires a GROUP clause.

(8 marks)

(b) **(i)** *Amend the SQL* you wrote for part (a) so that the output includes <u>ONLY</u> sports facilities <u>that have NO bookings</u>. The format of the output should be the same as that for part (a). *Hint: Use an Outer Join with RESTRICTION.*

(4 marks)

(i) *Identify* what the output should be when the SQL is executed for the data provided and *explain how* the SQL you wrote achieves this.

(3 marks)

(c) *Using* `UNION` *write the SQL* needed to *create* a `VIEW` called `FacilityUse` which has details about the number of bookings for each facility. You need to:

- Include a column `facility` which is the `facility description`;
- Include a column `bookings` which holds `the number of bookings` for that facility.

The SQL should be based on the SQL you wrote for parts (a) and (b) so you will need to ensure *a row is included* for each facility whether it has bookings or not.

(5 marks)

SECTION B

4. Suppose that the data in the database is as given in Q2 but that a `column teamrole` has been added to the `teammember table` and that this has been populated as follows:

STUDENTNO	FIRSTNAME	LASTNAME	TNO	TEAMROLE
1001	Weyand	Sabine	101	Captain
1031	Rabb	Dominic	103	Captain
1031	Rabb	Dominic	102	Player
1041	Johnson	Boris	103	(null)
1041	Johnson	Boris	102	(null)
1051	Riso	Stephanie	104	Captain
1051	Riso	Stephanie	101	Player

(a) Using `UNION` *write the SQL* to create the following output:

STUDENTNO	FIRSTNAME	LASTNAME	TNO	ROLE	TNAME
1001	Weyand	Sabine	101	IS CAPTAIN OF	Rowing
1031	Rabb	Dominic	102	IS A PLAYER ON	Boxing
1031	Rabb	Dominic	103	IS CAPTAIN OF	Archery
1041	Johnson	Boris	102	IS A PLAYER ON	Boxing
1041	Johnson	Boris	103	IS A PLAYER ON	Archery
1051	Riso	Stephanie	101	IS A PLAYER ON	Rowing
1051	Riso	Stephanie	104	IS CAPTAIN OF	Sailing

You need to ensure that `IS CAPTAIN OF` is output for Role when the column teamrole has the value Captain and that `IS A PLAYER ON` is output when the column teamrole has the value Player or has no value.

Hint: You need to join teammember to two other tables. You need to use RESTRICTION.

(9 marks)

(b) **(i)** Using `INTERSECT`, *write the SQL* to find the *students* who are *captain of* at least one team **and** *a player* for at least one other.

You need only output *studentno, firstname and lastname.*

(3 marks)

(ii) *Identify* clearly the output the SQL will produce when executed based on the data provided.

(2 marks)

(c) **(i)** *Amend the SQL* you created for part (a) to output for each student a `count of the number of teams on which they are captain or the number of teams on which they are a player` rather than the team name.

Sort the output `in order` of studentNo.

Hint: Use UNION and use a GROUP BY clause in each of the select statements.

(4 marks)

(ii) *Identify* clearly the output the SQL will produce when executed based on the data provided.

(2 marks)

SYNTAX TABLE

```
ALTER TABLE table column_clauses;
 column_clauses:
    ADD (column datatype [DEFAULT expr] [column_constraint(s)] [,…] )
    DROP COLUMN column    [CASCADE CONSTRAINTS]
    MODIFY column datatype [DEFAULT expr] [column_constraint(s)]
    RENAME COLUMN column TO new_name
ALTER TABLE table constraint_clause [,…];
constraint_clause:
    DROP PRIMARY KEY [CASCADE] [{KEEP|DROP} INDEX]
    DROP UNIQUE (column [,…]) [{KEEP|DROP} INDEX]
    DROP CONSTRAINT constraint [CASCADE]
    MODIFY CONSTRAINT constraint constrnt_state
    MODIFY PRIMARY KEY constrnt_state
    MODIFY UNIQUE (column [,…]) constrnt_state
    RENAME CONSTRAINT constraint TO new_name
COMMIT
CASE [ expression ]
    WHEN condition_1 THEN result_1
    WHEN condition_2 THEN result_2
    WHEN condition_n THEN result_n
    ELSE result
END
Conditions:=,>,<,>=,<=,<>, BETWEEN .. AND.., IN (list),IS NULL, IS NOT NULL,
LIKE
CREATE TABLE table ( column datatype [DEFAULT expr] [column_constraint(s)[,…]]
[,column datatype [,…]]
          [table_constraint [,…]])
Datatypes:    [CHAR [(n)] | VARCHAR2(n) | NUMBER [ n,p] | DATE | DATETIME]
Constraints: {[NOT NULL | UNIQUE | CHECK | PRIMARY KEY | FOREIGN KEY coltable1
         FOREIGN KEY REFERNECES table2(coltable2)]}
CREATE VIEW view_name AS
  SELECT columns
  FROM tables
  [WHERE conditions];
DELETE FROM tablename WHERE condition
DROP [TABLE tablename|DROP VIEW viewname]
INSERT INTO tablename (column-name-list) VALUES (data-value-list)
Logical operators:  AND, OR, NOT
ROLLBACK
SELECT [DISTINCT] select_list
      FROM table_list
         [WHERE conditions]
               [GROUP BY group_by_list]
                  [HAVING search_conditions]
                     [ORDER BY order_list [ASC | DESC]]
SELECT
      ... FROM table1 LEFT JOIN table2
         ON table1.field1 compopr table2.field2 | USING clause
      ... FROM table1 RIGHT JOIN table2
         ON table1.field1 compopr table2.field2 | USING clause
      ... FROM table1 INNER JOIN table2
         ON table1.field1 compopr table2.field2| USING clause
Key
   table1, table2   The tables from which records are combined.
   field1, field2   The fields to be joined.
   compopr          Any relational comparison operator: =  <  >  <=  >=  or  <>

SELECT expression1, expression2, ... expression_n
```

SYNTAX TABLE

```
FROM tables [WHERE conditions]
UNION
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions]
INTERSECT
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions]
MINUS
SELECT expression1, expression2, ... expression_n
FROM tables [WHERE conditions];

UPDATE tablename
[SET column-name= <data-value>] [WHERE condition]
```

ORACLE FUNCTIONS

Null Handling Functions: NVL, NVL2, NULLIF, COALESCE, CASE, DECODE.

Case Conversion functions - Accepts character input and returns a character value: UPPER, LOWER and INITCAP.

Character functions - Accepts character input and returns number or character value: CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM and REPLACE.

Date functions - Date arithmetic operations return date or numeric values: MONTHS_BETWEEN, ADD_MONTHS, NEXT_DAY, LAST_DAY, ROUND and TRUNC.

Group Functions: SUM([ALL | DISTINCT] expression); AVG([ALL | DISTINCT] expression); COUNT([ALL | DISTINCT] expression); COUNT(*); MAX(expression);MIN(expression)

Number fucntions - accept numerical input and return number output - ROUND, TRUNC, MOD

Formatting: TO_CHAR(value [, format_mask])| TO_DATE(string1 [, format_mask]) | TO_NUMBER(string1 [, format_mask] [, nls_language])

Formats: Year, year spelled out; YYYY 4-digit year; YY 2-digit year;
MM Month (01-12; JAN = 01); MON Abbreviated name of month; MONTH Name of month, padded with blanks to length of 9 characters;
WW Week of year (1-53) where week 1 starts on the first day of the year and continues to the seventh day of the year; W Week of month (1-5) where week 1 starts on the first day of the month and ends on the seventh;
D Day of week (1-7); DAY Name of day; DD Day of month (1-31);
HH Hour of day (1-12); MI Minute (0-59); SS Second (0-59);
9 Represents a number; 0 Forces a zero to be displayed; $ Places a floating dollar sign; U Local currency sign;
. Prints a decimal point; , Prints a comma as thousands indicator