DUBLIN INSTITUTE OF TECHNOLOGY

DT228 BSc. (Honours) Degree in Computer Science
DT282 BSc. (Honours) Degree in Computer Science (International)

Year 2

WINTER EXAMINATIONS 2015/2016

DATABASES I [CMPU2007]

Ms. D. Lawless
Dr. D. Lillis
Mr. K. Foley

Friday 8th January 9.30 a.m. – 11.30 a.m.

Answer ALL questions.

There is a syntax table on the last page to assist you.

Case Study #1 - Footie AFC

The following relational schema and interpretation will be used in subsequent questions:

clubMember(mID, fName, surname, email)
Manager(mgrID, mgrName, mgrEmail)
team(tNo, tName, mgrID)
teamMember (tNo, mID)
practiceSlots(pSlotNo, parkLocation, weekday, start_time, end_time)
practiceBooking(pSlotNo, tNo)
Note: underlined attributes indicate how each instance of a relation is identified.

Footie AFC is a football club based in Walkinstown, Dublin. The club has access to a number of playing pitches in Tymon Park.
Details of club members are stored in the clubMember table. Stored for each club member is a unique number (mID), their first name (fName) and surname (surname) and email address (email).
Details of managers are stored in the Manager table. Stored for each manager is a unique number (mgrID), their name (mgrName) and email address (mgrEmail).
Details of each team are stored in the team table. Stored for each team are its unique number (tNo), name (tName), and ID of the team manager who acts as the team manager (mgrID). Each team can only have one manager but each manager can manage many teams.
Each team has a number of members. The members for each team are stored on the teamMember table. Details stored are the number of the team (tNo) and the member ID of the member (mID). Each team can have many members. Each member can belong to only one team.
The club has a number of pitches booked in Tymon park for practice on selected days of the week and certain times on these days. Each practice slot available is stored in the practiceSlots table. Details stored are a unique number (pSlotNo), the location of the pitch in the park (parkLocation), the day of the week (weekday), the start of the timeslot it is available on that day (start_time) and the end time of that timeslot (end_time).
The team manager can book these slots for their teams. Details of practice bookings are stored in the practiceBooking table. Details stored are the number of the slot booked (pSlotNo) and the team number for which it is booked (tNo). Each slot can be booked by only one team but each team can book many practice slots.

1. **(a)** **(i)** Draw a *physical Entity Relationship diagram (ERD)* for the case study above showing the attributes for each entity and identifying clearly primary and foreign keys.

(14 marks)

(ii) Explain *what type of entity* practicebooking is. Explain clearly why it is necessary and what type of relationship it helps resolve when transforming from logical to physical design.

(6 marks)

(b) When creating the Footie AFC database the following requirements must be adhered to :

- all identifiers for all data must be numeric capable of storing up to 6 digits;
- all names and pitch locations must be alphanumeric capable of storing up to 30 characters and cannot be null;
- day of the week is stored as a numeric which can only take values 1 to 7;
- a manager must have an email address, team members do not need to provide one;
- start_time and end_time must be stored as a character value and values will be inserted in the format HH:MM;
- foreign key constraints must be named with the names of the tables involved and _fk e.g. table1_table_2_fk.

(i) Write the SQL to *Create* all the tables required by the Footie AFC database *including all constraints required.* Explain the order in which the tables need to be created and why.

(15 marks)

(ii) Explain clearly what *type of data integrity* each of the constraints you implemented (including primary keys) helps you to achieve.

(10 marks)

Question 1 continues on next page....

1. **(c)** Assuming that you have created tables for the Footie AFC in your schema.

Write the SQL statements required to *insert* the following data into the relevant tables and persist it.

Explain the order in which the data must be inserted and why an instruction is needed to persist it.

Team No	Team Name	Manager	Team Members
1	Blue	Mr. Murinho jmblue@gmail.com	J. Terry
			E. Hazard
2	Red	Mr. Klopp jkred@gmail.com	J. Milner
			E.Can
3	White	Mr. M. Pochettino	H.Kane
		mpwhite@gmail.com	hk@whiteafc.com

Practice Slot	Location	Day	Start Time	End Time	Booked by team
1	East 1	Monday	18.00	19.00	1
2	West 1	Monday	18.00	19.00	2
3	West 2	Tuesday	18.00	19.00	2
4	East 1	Monday	19.00	20.00	1

(15 marks)

2. **(a)** Write and explain the SQL needed to ensure that all club members have an email address which must be unique, without dropping and recreating the table. The email addresses of all club members are given in the table below:

Club Member	Email
J. Terry	jt@blueafc.com
E. Hazard	eh@blueafc.com
J. Milner	jm@redafc.com
E.Can	ec@redafc.com
H.Kane	hk@whiteafc.com

(6 marks)

(b) Write and explain the SQL needed to achieve the following, without dropping and recreating tables:

Practice slots are now booked by managers. Therefore each practice booking needs to record the manager ID, not the team no.

(10 marks)

(c) Write and explain the SQL needed to delete practice slot 3 but retain the detail of any bookings of that slot.

(4 marks)

3. **(a)** Assume that you have successfully inserted all the data give in Q1 into the relevant tables.

Write the SQL to achieve the following output and explain how the SQL statement achieves this.

(Hint: Use functions)

Team Membership
J. TERRY is on team BLUE No contact email available
E. HAZARD is on team BLUE No contact email available
J. MILNER is on team RED No contact email available
E. CAN is on team RED No contact email available
H. KANE is on team WHITE Contact at hk@whiteafc.com

(10 marks)

(b) Suppose there is a club member J. Cruyuff, member id 6 who is not currently a member of any team.

Write the SQL to achieve the output below and explain clearly what type of join is being used and how the SQL works.

FNAME	SURNAME	TNO
J.	Cruyuff	(null)
H.	Kane	3
J.	Milner	2
E.	Can	2
J.	Terry	1
E.	Hazard	1

(10 marks)

SYNTAX TABLE

```
ALTER TABLE [schema.]table column_clauses;
 column_clauses:
   ADD (column datatype [DEFAULT expr] [column_constraint(s)] [,...] )
   DROP COLUMN column
      [CASCADE CONSTRAINTS] [INVALIDATE] CHECKPOINT int
   MODIFY column datatype [DEFAULT expr] [column_constraint(s)]
   RENAME COLUMN column TO new_name
ALTER TABLE [schema.]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
CREATE TABLE table (
      column datatype [DEFAULT expr] [column_constraint(s)[,...]] [,column
datatype [,...]]
         [table_constraint [,...]])
COMMIT
DELETE FROM tablename WHERE condition
DROP [TABLE tablename|DROP VIEW viewname]
INSERT INTO tablename (column-name-list) VALUES (data-value-list)
ROLLBACK
SELECT [DISTINCT] select_list
      FROM table_list
         [WHERE conditions]
               [GROUP BY group_by_list]
                  [HAVING search_conditions]
                     [ORDER BY order_list [ASC | DESC] ]
```

SYNTAX TABLE

Conditions:=,>,<,>=,<=,<>, BETWEEN .. AND.., IN *(list)*,IS NULL, IS NOT NULL, LIKE

Logical operators: AND, OR, NOT

Set operations: UNION, MINUS, INTERSECT

```
CASE [ expression ]
   WHEN condition_1 THEN result_1
   WHEN condition_2 THEN result_2
   WHEN condition_n THEN result_n
   ELSE result
END
```

```
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 <>

UPDATE *tablename*

[SET *column-name= <data-value>*] [WHERE *condition*]

Column-definition* = *column-name **[CHAR [(*n*)] | VARCHAR2(*n*) | NUMBER [*n,p*] | DATE | DATETIME] {[NOT NULL | UNIQUE | PRIMARY KEY]}**

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)