DUBLIN INSTITUTE OF TECHNOLOGY

BSc. (Honours) Degree in Computer Science

Year 2

WINTER EXAMINATIONS 2014/2015

DATABASES I [CMPU2007]

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

FRIDAY 19TH DECEMBER 2014 9.30 A.M. - 11.30 A.M.

2 HOURS

Answer ALL questions.

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

Case Study #1 – D-Courier Company

The following relational schema (with keys underlined) and interpretation will be used in subsequent questions:

courier(cID, fName, surname, salary)
customer(cNo, cName, street, city, countyID, phone, email)
parcel(pNo, courierID, custNo, pdate, pvalue, deliverToID)
delivery(dID, delName, delStreet, delCity, delCountyID, delPhone)
county(countyID, countyName)

D-Courier company employs a number of couriers to collect parcels from customers and deliver each parcel to a specified name and address.

Details on the couriers are stored in the `courier` table. Associated with each courier is a unique number (`cID`), first name (`fName`) and surname (`surname`) and salary (`salary`).
Details on customers are stored in the `customer` table. Each customer has a unique number (`cNo`). Also stored are the customer's name, address, email address and phone number.
Information on parcels is stored in the `parcel` table. A unique code is assigned to each parcel (`number`) and the courier ID (`courierID`) is stored to indicate which courier is responsible for collection and delivery of the parcel. Also stored is the customer number (`custNo`), the date the job was entered (`pdate`), the value of the parcel (in Euros), and the ID of the company or person it is to be delivered to (`deliveryToID`).
Details on where to deliver the parcel are stored in the `delivery` table. The table holds a unique ID (`dID`), the delivery name, (`delName`), delivery address (`delStreet`, `delCity`, `delCountyID`) and phone number (`delPhone`). Each parcel will appear only once in the `delivery` table.
Note that one customer may have many parcels and that a courier may undertake many deliveries and that many parcels may be delivered to the same person and address.

1. **(a)** Explain the following concepts:

- Entity
- Attribute
- Primary Key
- Foreign Key

Illustrate your answer by drawing an *Entity Relationship diagram (ERD)* for the case study above.

(20 marks)

(b) In the D-Courier database, all identifiers for all data must be numeric capable of storing up to 6 digits; all names, addresses and phone numbers must be alphanumeric capable of storing up to 30 characters; all monetary fields (value and salary) must be capable of storing values up to 99,999.99.

Write the SQL to *Create* all the tables including all *primary* and *foreign* key constraints required.

Outline the order in which the tables need to be created and explain why.

(15 marks)

(c) Assuming that you have created tables for the D-Courier company in your schema and you have populated them with relevant data.

(i) Write the SQL statements required to *insert* the following data into the Courier table and persist it:

cID	fName	surname	Salary
1003	Slim	Shady	20,000
1099	Ruby	Tuesday	20,000
2123	Daisy	Duke	18,000
3456	Johnny	Bride	17,000

(5 marks)

(ii) Write a SQL statement, using a JOIN, to *retrieve* the name and addresses of all customers who live in Cork County, including the name of the county in the output, given that the countyID for Cork is 20.

Identify the type of join used and explain how the data is retrieved from the tables concerned, explaining clearly how you differentiate fields with the same names in different tables.

(10 marks)

(iii) Amend the SQL for part (ii) to provide the *average* value of all parcels for each county in which customers exist.

(5 marks)

(iv) Write a SQL statement to *set the value of courierID* of parcel number 20 to be the same as the courierID on parcel number 99.

Explain how this SQL statement works.

(5 marks)

2. Explain clearly the difference between a *LEFT OUTER JOIN* and *RIGHT OUTER JOIN*.

Illustrate your answer by writing the SQL to achieve the following:

- Retrieve details of all counties in Ireland and if customers exist in that county, the city associated with customer who lives in the county, otherwise show null for this field. Sort the output in descending order.

- The names of all those who accepted delivery for a parcel and for those whose ID matches an existing customer show the email address, otherwise show null. Sort the output in ascending order.

(20 marks)

3. Explain clearly the difference between *REFERENTIAL INTEGRITY* and *VALUE INTEGRITY*.

Illustrate your answer by writing SQL using the *ALTER* statement to achieve the following and explain clearly what would need to happen if any existing data violated these constraints in order to successfully establish these constraints.

- Add a field DeliveredStatus to the Parcel table which will indicates whether a parcel has been delivered successfully, can only accept values of Y or N and which should have a default value of N.
- Amend the relevant attribute of the Parcel table to facilitate parcels of values up to 999999.99.
- Add an attribute to the delivery table to store an email address which cannot be null and should be a unique value.
- Add a referential constraint to ensure that all deliveryIDs in the delivery table exist as customers.

(20 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] ]
Conditions:=,>,<,>=,<=,<>, BETWEEN .. AND.., IN (list),IS NULL, LIKE
Logical operators:  AND, OR, NOT
Set operations:  UNION, MINUS, INTERSECT
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)] | VARCHAR(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)
```