****

**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 1.00 P.M. - 3.00 P.M.

2 HOURS

**Answer ALL questions.**

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

**RATIONALE FOR PAPER**

On completion of CMPU2007 DATABASES I module, the module specification states that the learner will be able to:

1. Describe and justify the rationale behind the use of relational database management systems.   
2. Demonstrate an understanding of the desirable features of a database management system and how they are achieved.   
3. Identify and distinguish between data and meta-data, and the concepts of keys.   
4. Design a data model suited to a business application and implements it in a relational database.   
5. Define tables and views with appropriate constraints to ensure data integrity and relational integrity.   
6. Manipulate the data in a relational database using DDL and DML aspects of SQL.   
7. Apply knowledge of SQL to real-world query problems.

This examination is part of an overall assessment strategy of which 50% is contributed by this end of semester examination and 50% contributed by continuous assessment activities.

The student is required to answer all questions to make this a true standardised assessment of their knowledge and skills under time pressure to facilitate direct comparison of student performance.

The examination assesses aspects of all the learning outcomes stated above however given the time constraints available the nature of problems presented are not of the scale of those tackled during the continuous assessment.

As the learning outcomes require demonstration of understanding, and this requires knowledge of both theory and practical skills, some questions require the student to present their knowledge and understanding of key concepts and practices together with an illustration for a given scenario/example.

The companion continuous assessment requires students to undertake a small scale design and implementation exercise under time constraints, a larger independent project with a demonstration of ability to manipulate and alter data and structures and a task to develop questions to test their own and the knowledge of their peers of the required content.

|  |
| --- |
| **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) |
| **Sample Marking Scheme:**  **4 x 2 marks (8 marks)** for correct explanation of entity, attribute, primary key, foreign keys  **4 x 2 (8 marks)** for correct identification of examples of each concept  **4 marks**  for correct placement and drawing of attributes, keys and relationships  **Sample Answer:**  Entity: THING or OBJECT of interest to the organization being modeling  Attribute: Descriptive properties possessed by all members of an entity set. Pieces of data we want to store about an entity  Primary Key: Instances are identified through keys. The primary key to an entity is an attribute or set of attributes that uniquely identify each occurrence of the entity. Must be unique for each occurrence of an entity. Must always have a value  Foreign Keys: When one table needs to be related to another table, you must include a common field. The common field will be the primary key in one table. The common field is referred to as the foreign key. The foreign key in a table can then be used as a primary key to access the record in the related table when needed. A foreign key value must match an existing value in the parent table or be NULL. | | |

|  |  |  |
| --- | --- | --- |
| **1.** | **(b)** | All identifiers for all data must be numeric and be capable of storing up to 6 digits in the D-Courier database. All names, addresses and phone numbers must be alphanumeric and be 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) |
| **Sample Marking Scheme:**  **5 x 2 (10 marks)** for correctly formatted create statements with correct datatypes and primary keys  **3 marks** for correct foreign keys  **2 marks** for correct creation order and explanation  **Sample SQL:**  create table courier(  cID number(6) primary key,  fName varchar2(30),  surname varchar2(30),  salary number(7,2));  create table county(  countyID number(6) primary key,  countyName varchar2(30));  create table customer(  cNo number(6) primary key,  cName varchar2(30),  street varchar2(30),  city varchar2(30),  countyID number(6),  constraint cust\_county\_fk foreign key (countyID) references county(countyID),  phone varchar2(30),  email varchar2(30));  create table delivery(  dID number (6) primary key,  delName varchar2(30),  delStreet varchar2(30),  delCity varchar2(30),  delCountyID number(6),  constraint del\_county\_fk foreign key (delCountyID) references county(countyID),  delPhone varchar2(30)) ;  create table parcel (  pNo Number(6) primary key,  courierID number(6),  constraint p\_courier\_fk foreign key (courierID) references courier(cID),  custNo number(6),  constraint p\_cust\_fk foreign key (custNo) references customer(CNo),  pdate date,  pvalue number(7,2),  deliverToID number(6),  constraint p\_deliver\_fk foreign key (deliverToID) references delivery(dID)) ; | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **1.** | **(c)** | Assuming that you have created tables for the D-Courier company in your schema and you have populated them with relevant data.   1. 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) |
| **Sample Marking Scheme:**  **4 x 1 (4 marks)** for correctly formatted insert statements with correct data  **1 mark** for commit.  insert into courier values (1003, 'Slim', 'Shady', 20000.00);  insert into courier values (1099, 'Ruby', 'Tuesday', 20000.00);  insert into courier values (2123, 'Daisy', 'Duke', 18000.00);  insert into courier values (3456, 'Johnny', 'Bride', 17000.00);  commit; | | |
|  |  |  |
|  |  | 1. 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) |
| **Sample Marking Scheme:**  **1 marks** for including correct fields  **2 marks**  for correct where clause  **3 marks**  for joining correct tables on correct fields  **3 marks**  for explaining join  **1 marks**  for explaining how to differentiate fields by prefixing with table name  **Sample Answer:**  The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.  **Sample SQL:**  select cname, STREET, CITY,county.COUNTYNAME from customer inner join county on customer.COUNTYID = county.COUNTYID and customer.countyID=20;  select cname, STREET, CITY,county.COUNTYNAME from customer join county on customer.COUNTYID = county.COUNTYID and customer.countyID=20; | | |
| **1.** | **(c)** | 1. Amend the SQL for part (ii) to provide the *average* value of all values for each county in which customers exist.   (5 marks) |
| **Sample Marking Scheme:**  **1 mark** for avg on correct field  **2 marks** for group by clause on correct field  **2 marks**  for correct join on correct tables  **Sample SQL:**  select countyid, avg(pvalue) from parcel join customer on parcel.CUSTNO =customer.CNO group by countyid; | | |
|  |  |  |
| **1.** | **(c)** | 1. 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.   (5 marks) |
| **Sample Marking Scheme:**  **3 marks** for correctly using update with set and where  **2 marks**  for correct select where clause  **Sample SQL:**  update parcel set courierID =(select courierID from parcel where pno=99) where pno=20; | | |
|  |  |  |

|  |  |
| --- | --- |
| **2.** | Explain clearly the difference between a *LEFT OUTER JOIN*, *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) |
| **Sample Marking Scheme:**  **2 x 3 marks (6 marks)** for correctly explaining each type of join  **2 x 4 marks (8 marks)**  for correct syntax for joins joining on correct fields on correct tables  **2 marks** for including correct fields in select  **2 marks** for correct sort syntax.  **2 marks** for choosing correct join for information requirement stated  **Sample Answer:**  The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.  The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.  **Sample SQL:**  select county.countyid, countyname, customer.city from county left join customer on customer.COUNTYID = county.COUNTYIDselect cno, cname, pno from parcel right outer join customer on cno=parcel.custno order by parcel.custno desc;  select delivery.delname, customer.EMAIL from customer right outer join delivery on customer.CNO=delivery.did; | |

|  |  |
| --- | --- |
| **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) |
| **Sample Marking Scheme:**  **2 x 2 marks (4 marks)** for correctly explaining referential integrity and data integrity  **4 x 2 marks (8 marks)**  for correct SQL for each requirement  **4 x 2 marks (8 marks)** for correctly explaining not null, unique, check and foreign key  **Sample Answer:**  Referential Integrity: ensures relationships between tables remain consistent. When one table has a foreign key to another table, you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table. Can also ensure cascading deletions to preserve integrity.  Value Integrity: rules to ensure consistency and accuracy (validity) of the data stored in tables according to requirements of business being modelled. Specification of datatype and constraints. Ensure data cannot be inserted that is not valid.  **Sample SQL:**  --Add a field DeliveredStatus  alter table parcel Add (DeliveredStatus char(1) DEFAULT 'N' check (DeliveredStatus in ('Y','N')));  --Amend the relevant attribute of the Parcel table to facilitate parcels of values up to 999999.99.  alter table parcel modify (pvalue number(8,2));  --Add an attribute to the delivery table to store an email address which cannot be null and should be a unique value.  alter table delivery add (delemail varchar2(30) not null unique);  --Add a referential constraint to ensure that all deliveryIDs in the delivery table exist as customers.  alter table delivery add constraint del\_cust\_fk foreign key (did) references customer(cno); | |