**COURSE CCPS** **510**

**DATABASE SYSTEMS (DBMS) I**

**ASSIGMENT-I**

|  |  |
| --- | --- |
| **INSTRUCTOR:** | Ersan Cam |
| **INSTRUCTOR E-MAIL:** | [ersan.cam@ryerson.ca](mailto:ersan.cam@ryerson.ca) |

**Please place all your answers and screen shots of your results into this document. Once you finish working on this file, please email me.**

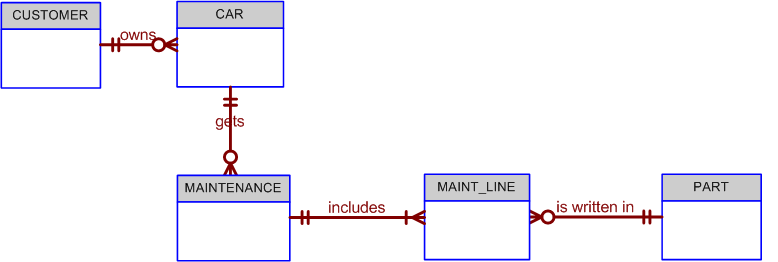
This Assignment has different parts each represent different Labs (Ideal time to solve these are in dedicated Lab times of the course)

**Lab Section A:**

**Choose either Task 1-1 or Task 1-2 (Only one of them)**

**Task1-1)**  
**Suppose you are working within the framework of the conceptual model in Figure 1 below.**

**Figure 1 The Conceptual Model Car Maintenance Services**



**Given the conceptual model in Figure 1:**

* 1. **Write the business rules that are reflected in it.**

Even a simple ERD such as the one shown in Figure Q4.5 is based on many business rules. Make sure that each business rule is written on a separate line and that all of its details are spelled out. In this case, the business rules are derived from the ERD in a “reverse-engineering” procedure designed to document the database design. In a real world database design situation, the ERD is generated on the basis of business rules that are written before the first entity box is drawn. (Remember that the business rules are derived from a carefully and precisely written description of operations.)

Here are business rules:

1.  **A customer can own many cars, but each car can have only one customer.**

2.  **A car can get many maintenances, but each maintenance can have only one car.**

3.  **A maintenance includes one or more maintenance lines, but a maintenance line can have only one maintenance.**

4.  **A maintenance line is written in one part only but a part can have many maintenance lines.**

5.  **A customer can have many maintenances, but a maintenance can only contain one customer.**

6.  **A customer can have many maintenance lines, but each maintenance line can only have one customer.**

Task 1-2

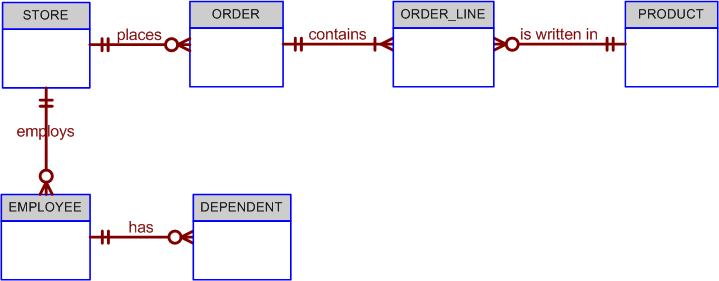
Use the following business rules to create a Crow’s Foot ERD. Write all appropriate connectivity and cardinalities in the ERD.

* + 1. A department employs many employees, but each employee is employed by one department.
    2. Some employees, known as “rovers,” are not assigned to any department.
    3. A division operates many departments, but each department is operated by one division.
    4. An employee may be assigned many projects, and a project may have many employees assigned to it.
    5. A project must have at least one employee assigned to it.
    6. One of the employees manages each department, and each department is managed by only one employee.
    7. One of the employees runs each division, and each division is run by only one employee.

Task 2)

Below questions is based on the ERD in Figure 2.

**FIGURE 2 - The ERD of**



2-a)

Fill Cardinalities. I have left 2 of them on the diagram for you as hint. You can fill rest of 8

**The Cardinalities**

Diagram

Description automatically generated

P.S: You can print and write on the paper and take picture and paste here or take this above picture as screen shot and edit on it or use draw.io free cloud tool to complete ERD diagrams.

2-b) Write at least 6 Business Rules by analyzing ERD diagram above

**1) A store can place many orders, but each order can only have one store.**

**2) Each order can contain one or more order lines, but each order line can only be from one order.**

**3) Each order line is written in one product only, but a product can be from many order lines.**

**4) A store can employ many employees, but an employee can only belong to one store.**

**5) An employee can have many dependents, but a dependent belongs to only one employee.**

**6) A store can have many employee dependents, but an employee dependent can only belong to one store.**

This following section (Lab B) is for Physical Database Design practices to build database tables and add constraints to apply business rules by using DDL (Data Definition Language ) of SQL.

**Lab Section B: DDL**

Graphical user interface, text

Description automatically generated

Text

Description automatically generated with low confidence

Graphical user interface, text, application

Description automatically generated

**1)**

**CREATE TABLE store\_reps**

**( rep\_ID NUMBER(5) PRIMARY KEY,**

**last VARCHAR2 (15),**

**first VARCHAR2(10),**

**comm CHAR(1) DEFAULT 'Y');**

Graphical user interface, text, application

Description automatically generated

**2)**

**ALTER TABLE store\_reps**

**MODIFY last VARCHAR2 (15) NOT NULL;**

**ALTER TABLE store\_reps**

**MODIFY first VARCHAR2(10) NOT NULL;**

Graphical user interface, application

Description automatically generated

**3)**

**ALTER TABLE store\_reps**

**ADD CONSTRAINT CHECK\_COMM\_Y\_OR\_N CHECK (comm = 'Y' OR comm = 'N');**

Graphical user interface, text, application, Word

Description automatically generated

**4)**

**ALTER TABLE store\_reps**

**ADD Base\_salary NUMBER(7,2);**

**ALTER TABLE store\_reps**

**ADD CONSTRAINT SALARY\_CHECK CHECK (Base\_salary > 0);**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**5)**

**CREATE TABLE BOOK\_STORES**

**( Store\_ID NUMBER(8) PRIMARY KEY,**

**Name VARCHAR2(30) NOT NULL UNIQUE,**

**Contact VARCHAR2(30),**

**Rep\_ID VARCHAR2(5));**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**6)**

**ALTER TABLE BOOK\_STORES**

**MODIFY Rep\_ID NUMBER(5);**

**ALTER TABLE Book\_STORES**

**ADD CONSTRAINT fk\_rep\_id FOREIGN KEY (Rep\_ID) REFERENCES store\_reps(rep\_ID);**

Graphical user interface, text, application

Description automatically generated

Graphical user interface, application

Description automatically generated

**7)**

**ALTER TABLE BOOK\_STORES**

**ADD FOREIGN KEY (Rep\_ID) REFERENCES store\_reps(rep\_ID)**

**ON DELETE CASCADE;**

**8)**

**CREATE TABLE REP\_CONTRACTS**

**( Store\_ID NUMBER(8),**

**Name NUMBER (5) NOT NULL UNIQUE,**

**Quarter CHAR(3),**

**Rep\_ID NUMBER(5),**

**PRIMARY KEY (Store\_ID, Quarter, Rep\_ID),**

**FOREIGN KEY (Store\_ID) REFERENCES BOOK\_STORES(Store\_ID),**

**FOREIGN KEY (Rep\_ID) REFERENCES store\_reps(rep\_ID)**

**);**

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, application

Description automatically generated

**9)**

**SELECT \***

**FROM USER\_CONSTRAINTS**

**WHERE table\_name = 'STORE\_REPS';**

Graphical user interface, table

Description automatically generated

**10)**

**alter table store\_reps disable constraint SALARY\_CHECK;**

**alter table store\_reps enable constraint SALARY\_CHECK;**

This following section (Lab C) is for practice SELECT command to query data.

Lab Section C) SELECT queries

**Develop SQL code for below questions**

**1- (Please Use Dream Home Database sample Data set)**

1. Find the number of staff working in each branch and the sum(Total) of their salaries

1. Provide a report for all of employees( staff) whose salary is greater than the Dream Home company’s average salary, and also calculate by how much their salary is greater than the average salary?

1. For each Branch office , list the staff numbers and names of staff who manage properties and the properties that they manage

Hint: Join Staff and PropertyForRent

1. List all properties and any branch offices that are in the same city

Hint: outer join between Branch and PropertyForRent

1. Increase all assistant staff salary by %10 UPDATE)
2. For staff name called Ann Beech , write an update command to promote her as Manager and increase her salary to 14000 and also move her to same branch as John White (SL21)

**2- Use HumanResources Database for below questions**

1. Display department name, manager name, and city.
2. Display country name, city, and department name.
3. Display employee name if the employee joined before his manager.

d.

A picture containing graphical user interface

Description automatically generated

Hint : Use Like with % or TO\_CHAR(Hire\_Date,’YYYY’) = ‘1994’

**e.**

A picture containing diagram

Description automatically generated

**1a)**

**SELECT COUNT(\*) as num\_of\_staff, SUM(salary)**

**FROM STAFF**

**GROUP BY BRANCHNO;**

Graphical user interface, application

Description automatically generated with medium confidence

**1b)**

**SELECT STAFFNO, salary - (SELECT AVG(salary) FROM STAFF) as above\_average\_salary**

**FROM STAFF**

**WHERE (salary - (SELECT AVG(salary) FROM STAFF)) > 0;**

A picture containing graphical user interface

Description automatically generated

**1c)**

**SELECT staff.branchno, staff.staffno, staff.fname, staff.lname, propertyforrent.street**

**FROM staff**

**INNER JOIN propertyforrent ON staff.staffno = propertyforrent.staffno**

**order by staff.branchno;**

Graphical user interface, text, application

Description automatically generated

**1d)**

**SELECT branch.branchno, propertyforrent.propertyno, branch.city**

**FROM branch**

**LEFT OUTER JOIN propertyforrent ON branch.city = propertyforrent.city**

**order by branch.branchno;**

Graphical user interface, text, application, email

Description automatically generated

**1e)**

**UPDATE staff**

**SET salary = salary \* 1.1**

**WHERE position = 'Assistant';**

**2a)**

**SELECT hr\_departments.department\_name, (hr\_employees.first\_name || ' '|| hr\_employees.last\_name) as Manager\_name, hr\_locations.city**

**FROM hr\_departments**

**LEFT OUTER JOIN hr\_employees ON hr\_departments.manager\_id = hr\_employees.employee\_id**

**LEFT OUTER JOIN hr\_locations ON hr\_departments.location\_id = hr\_locations.location\_id;**

Graphical user interface, application, Word

Description automatically generated

**2b)**

**SELECT hr\_countries.country\_name, hr\_locations.city, hr\_departments.department\_name**

**FROM hr\_countries**

**LEFT OUTER JOIN hr\_locations ON hr\_countries.country\_id = hr\_locations.country\_id**

**LEFT OUTER JOIN hr\_departments ON hr\_locations.location\_id = hr\_departments.location\_id**

**order by hr\_countries.country\_name;**

Graphical user interface, text, application, email

Description automatically generated

**2c)**

**SELECT**

**(employee.first\_name || ' '|| employee.last\_name) as employee\_name,**

**employee.hire\_date,**

**employee.Manager\_Id,**

**managerr.hire\_date as manager\_hire\_date**

**FROM hr\_employees employee**

**JOIN hr\_employees managerr**

**ON managerr.Manager\_Id = employee.employee\_id**

**Where (employee.manager\_id is not null) and (managerr.hire\_date - employee.hire\_date) > 0;**

Graphical user interface, text, application

Description automatically generated

**2d)**

**SELECT last\_name, hire\_date**

**From hr\_employees**

**where TO\_CHAR(Hire\_Date,'YYYY') = '1994';**

Graphical user interface, application

Description automatically generated

**2e)**

**SELECT last\_name, job\_id**

**From hr\_employees**

**where manager\_id is not null;**

Graphical user interface, text, application, email

Description automatically generated

**Lab Section D) Short Essay writings**

1. Define Entity Integrity and Referential Integrity. Talk a bout their similarities (if there is any) and most importantly differences. Roles of these two types of integrity rules? Why are entity **integrity and referential integrity important in a** database? How we implement each of thee integrity rules?

a)Discuss the difference between a composite key and a composite attribute. How

would each be indicated in an ERD?

b) What two courses of action are available to a designer who encounters a multivalued

attribute?

c) What are multivalued attributes, and how can they be handled within the database

design?

**3-**

a)What three data anomalies are likely to be the result of data redundancy? How can

such anomalies be eliminated?

b) Define and discuss the concept of transitive dependency

c) List of all Identifier (Key) types and explain each of them with two lines of your own words. Also discuss similarities, differences and relations between the Keys if there are any

**1)**

**Entity integrity states that each entry of a table has a primary key that is unique and not null. Reference integrity refers to all foreign keys in a table must match a specified primary key in the parent’s table, or it can be a null entry.**

**They are both concerned with obeying rules for a key entry (PK or FK). Their differences are, reference integrity can have keys that are null and not unique, but entity integrity needs to have keys that are the opposite (can’t be null and must be unique). Also, entity integrity is about the validity of an entry (one table), but reference integrity is about validity of the relationship between two tables.**

**Role of entity integrity: uniquely identify each entry by assigning a primary key (not null and unique), and in the process help to establish referential integrity (if any).**

**Role of referential integrity: Establish a valid relationship between two tables by linking each foreign key of the entry to a valid primary key of another table.**

**Entity integrity is important because it ensures there is no duplicate record in the database (the creation of the primary key helps us to do that).**

**Referential integrity is important to make sure the data in a database remains accurate and consistent. (It does that by linking foreign key to a primary key, so any potential changes to that PK will alert the corresponding FK entry as well)**

**How to implement Entity integrity: each entry has a primary key that is not null and unique.**

**How to implement referential integrity: each entry has a foreign key that matches a primary key from the parent’s table.**

**2a)**

**A composite key is a primary key that consists of two or more attributes. A composite attribute is a single attribute that can be further divided into additional attributes. In Chen’s model of ERD, the composite key is denoted by underlining all its attributes; And a composite attribute is denoted by an oval comprising of ovals (the first oval is the main attribute, and the linking ovals are the subdivided attributes).**

**2b)**

**solution 1 of multi-value attribute: Create an additional attribute for each multi-value attribute component. Those newly created attributes will be a single value.**

**solution 2 of multi-value attribute: Create a new entity with each attribute defining the components of the multi-value attribute. For example, if an employee has multiple children in the children attribute, create a “children” table with “name” as an attribute, then put all the children in that column and assign an employee ID to each child.**

**2c)**

**Multi-value attributes are attributes that have more than one value. How they can be handled is mentioned in 2b. It is represented as a double oval in Chen’s model**of **ERD.**

**3a)**

**The three anomalies that arise from data redundancy are insert anomaly, delete anomaly, and update anomaly. These anomalies can be eliminated by normalizing the table to the second form (assuming the table is in the first form already). The second form eliminates duplication by creating a new table for each partial dependency within the table. Then it creates a bridge table that connects all the newly created tables.**

**3b)**

**Transitive dependency: a non-PK attribute is the determinant of some other non-PK attributes. Primary key has an indirect relationship with a non-PK attribute that relies on another non-PK attribute. The 3rd normal form handles transitive dependency by creating a new table for each transitive dependency and using the original non-PK attribute determinant as its new primary key.**

**3c)**

**Single Key: key with one attribute**

**Composite Key: key with multiple attributes**

**Super Key: Combination of all attributes that uniquely identify an entry. A super key may be a single key or a composite key.**

**Candidate Key: Similar to a super key, but It’s a minimal set (no redundant attribute – an attribute that doesn’t contribute to the uniqueness of an entry) A candidate key is a super key, but a super key might not be a candidate key because super key allows redundant attributes.**

**Primary Key: It is a chosen candidate key that will uniquely identify an entry (there can be multiple candidate keys to choose from). It must be unique and not null. Primary key is a candidate key and is also a super key.**

**Foreign Key: Foreign key links to a primary key or a unique key in the parent’s table. It is there to ensure referential integrity to keep data consistent. Foreign key is usually not a good candidate for primary key because they can be not unique.**

**Secondary key: it is used as an alternative to the primary key to access data when PK is not known. Sometimes more than one secondary key is needed to uniquely identify an entry. A secondary key is not a primary key because it can be not unique, it is also not a foreign key because it doesn’t reference a key from another table.**

**Rubric for Lab Section A, B and C**

Graphical user interface, text, application

Description automatically generated

**Rubric for Lab Section D (Essay Writing)**

**Table

Description automatically generated**