**KLEF**

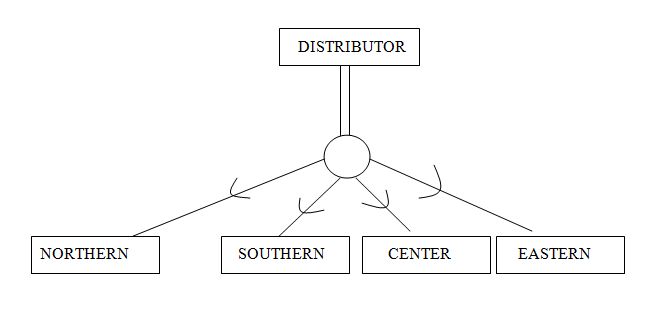
**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**19CS2108A DATABASE MANAGEMENT SYSTEMS**

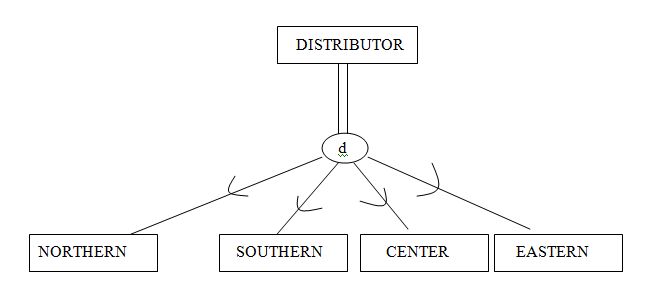
**TEST 1 KEY & SCHEME OF EVALUATION**

**Q.1. The entity type DISTRIBUTOR has four subclasses: NORTHERN, SOUTHERN, CENTRAL and EASTERN. Design EER diagram segment for each of the following situations: i. At a given time, a DISTRIBUTOR must be exactly one of these subclasses. ii. A DISTRIBUTOR may or may not be one of these subclasses. However, a DISTRIBUTOR who is one of these subclasses cannot at the same time be one of the other subclasses. iii. A DISTRIBUTOR may or may not be one of these subclasses. On the other hand, a DISTRIBUTOR may be any two or even three of these subclasses at the same time. iv. At a given time a DISTRIBUTOR must be at least one of these subclasses**.

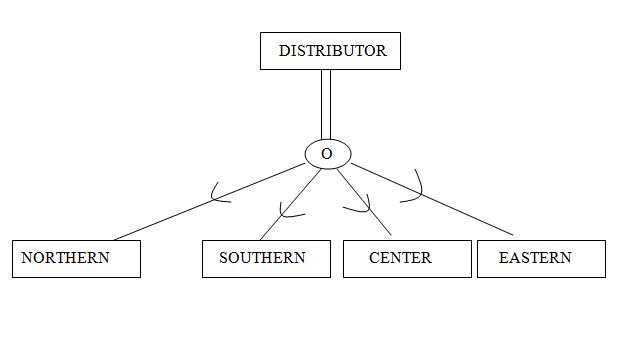
i. **At a given time, a DISTRIBUTOR must be exactly one of these subclasses.[1.5M]**



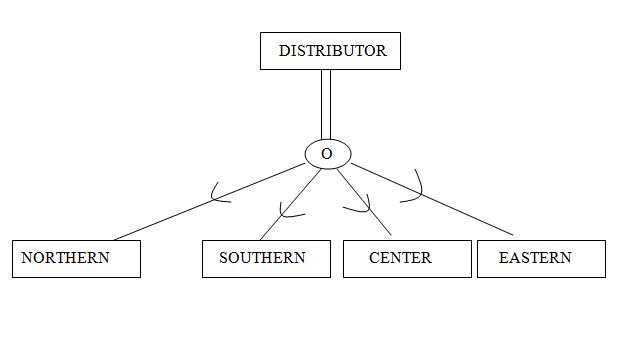
**ii.A DISTRIBUTOR may or may not be one of these subclasses. However, a DISTRIBUTOR who is one of these subclasses cannot at the same time be one of the other subclasses.** **[1M]**



1. **A DISTRIBUTOR may or may not be one of these subclasses. On the other hand, a DISTRIBUTOR may be any two or even three of these subclasses at the same time.** **[1M]**



**IV.At a given time a DISTRIBUTOR must be at least one of these subclasses.[1M]**



1. **At a given time, a DISTRIBUTOR must be exactly one of these subclasses. [1.5M]**
2. **A DISTRIBUTOR may or may not be one of these subclasses. However, a DISTRIBUTOR who is one of these subclasses cannot at the same time be one of the other subclasses. [1M]**

**iii. A DISTRIBUTOR may or may not be one of these subclasses. On the other hand, a DISTRIBUTOR may be any two or even three of these subclasses at the same time. [1M]**

1. **At a given time a DISTRIBUTOR must be at least one of these subclasses**.**[1M]**

**Q.2. What are the responsibilities of a DBA? If we assume that the DBA is never interested in running his or her own queries, does the DBA still need to understand query optimization? Why?**

**1-Schema definition**:

DBA prepares the database schema through implement set of data definition in DDL

**2- Storage body and define the access method**

**3- Schema and physical-organization modification:**

The DBA execute changes on the schema and physical organization to invert all the needs changing of the organization or change the organization physical to progress performance.

4- **Granting of authorization for data access.**

The DBA can organize any part of data base allow to users can access by agree to give various kinds of authorization and keep the information in private system structure that the DB system confer whenever try anyone access to the data in system.

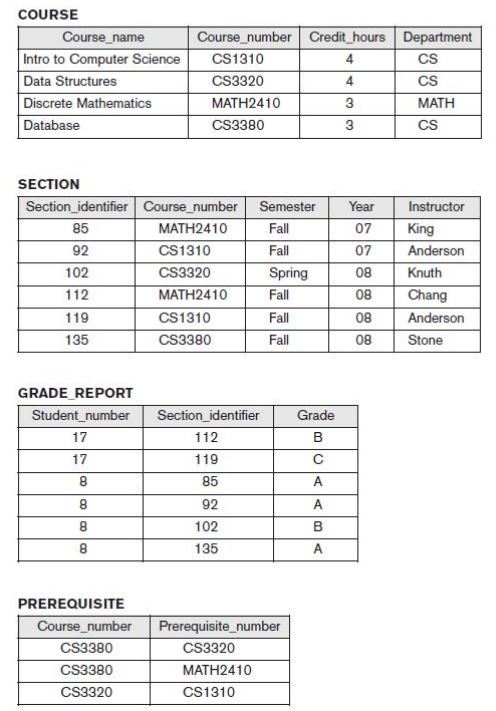
5- **Routine maintenance.**

The activities of Routine maintenance like support Periodically Up the DB to prevent loss data, and making sure if free disk space enough and control chances of work on DB and making sure the performance.

In most environments, the database administrator is expected to help tune poorly performing queries. After all, the DBA is the expert and is responsible for overall database performance. Indeed, tuning a query to eliminate excessive disk I/O or CPU processing will generally buy more in performance than you can normally get by tuning the System Global Area (SGA) or by optimizing the placement of data files on disk. The optimizer is that portion of the kernel that evaluates the SQL statement and determines the optimal way to retrieve the desired result set.

**Any 4 responsibilities**  **[4.5M]**

**Q.3. Specify all the relationships among the records of the database.**



1) Each SECTION record is related to a COURSE record.

2) Each GRADE\_REPORT record is related to one STUDENT record and one SECTION

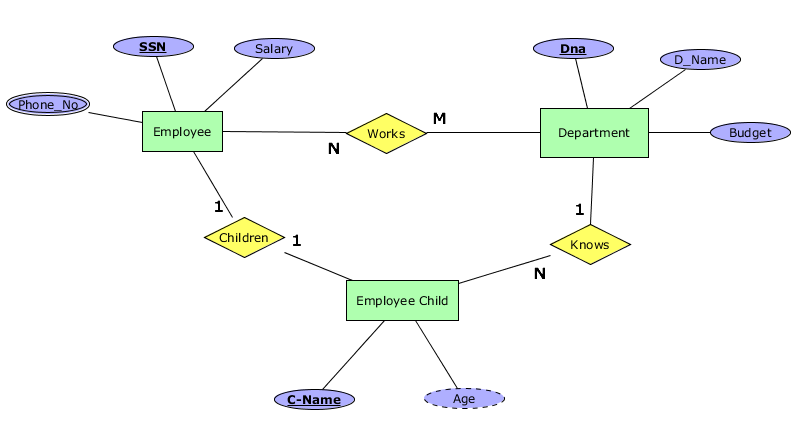
record.

3) Each PREREQUISITE record relates two COURSE records: one in the role of a course

and the other in the role of a prerequisite to that course.

**Mention 3 Relationships**  **[8M]**

**Q.4. A company database needs to store information about employees (identified by ssn, with salary and phone as attributes), departments (identified by dna, with dname and budget as attributes), and children of employees (with name and age as attributes). Employees work in departments; each department is managed by an employee; a child must be identified uniquely by name when the parent (who is an employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company. Draw an ER diagram that captures this information.**



**Diagram**  **[4M]**

**Constraints [4M]**

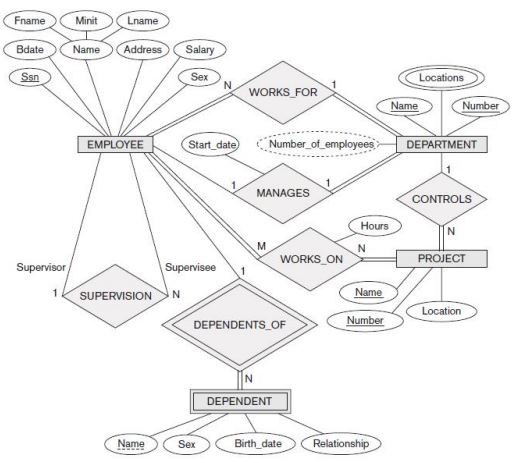
**Q.5.Computer Sciences Department frequent fliers have been complaining to Dane County Airport officials about the poor organization at the airport. As a result, the officials decided that all information related to the airport should be organized using a DBMS, and you have been hired to design the database. Your first task is to organize the information about all the airplanes stationed and maintainable at the airport. The relevant information is as follows: Every airplane has a registration number, and each airplane is of a specific model.• The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-lO) and has a capacity and a weight.• A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.• Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.• Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recentexam.• All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by a social security number.**

**Diagram**  **[7M]**

**Constraints [5.5M]**



**Q.6.Design Relational Model for the below mentioned ER diagram.**

**Employee**(ssn, Bdate, Fname ,Lname, Minit, Address,Salary,Sex)

**Department**(Name, Location, Number, Number of employees)

**Projec**t(Name, Number,Location)

**Dependent**(Name,sex, ssn,Birth\_date,relationship)

**Works\_for**(ssn,name,number)

**Manages**(ssn, start\_date, Name, number)

**Works\_on**(ssn,name,number,hours)

**Dependentsof**(ssn,name)

**Tables [6.5 M]**

**Insert records according to mentioned constraints [6 M]**

**Q.7. Consider the following relations for a database that keeps track of automobile sales in a car dealership (OPTION refers to some optional equipment installed on an automobile):CAR(Serial\_no, Model, Manufacturer, Price)OPTION(Serial\_no, Option\_name, Price)SALE(Salesperson\_id, Serial\_no, Date, Sale\_price)SALESPERSON(Salesperson\_id, Name, Phone)First, specify the foreign keys for this schema, stating any assumptions you make. Next, populate the relations with a few sample tuples, and then give an example of an insertion in the SALE and SALESPERSON relations that violates the referential integrity constraints and of another insertion that does not.**

CAR(Serial\_no)🡪FOREIGN KEY REFERENCE OPTION(Serial\_no)

SALE(Salesperson\_id)🡪FOREIGN KEY SALESPERSON(Salesperson\_id)

**INSERTING VALUES**

**SHOWING VIOLATION IN INTEGRITY CONSTRAINTS**

INSERT INTO CAR(1,’INNOVA1’,’INNOVA’,1000000)

INSERT INTO CAR(1,’INNOVA1’,’INNOVA’,1000000) 🡪violating integrity constraint because CAR SERIAL NUMBER IS PRIMARY KEY WHICH SHOULD BE UNIQUE

1. Consider the following example. It is natural to require that the did field of Works should be a foreign key, and refer to Dept.

CREATE TABLE Works ( eid INTEGER NOT NULL , did INTEGER NOT NULL , pcttime INTEGER, PRIMARY KEY (eid, did), UNIQUE (eid), FOREIGN KEY (did) REFERENCES Dept )

When a user attempts to delete a Dept tuple, There are four options:

Also delete all Works tuples that refer to it.

Disallow the deletion of the Dept tuple if some Works tuple refers to it.

For every Works tuple that refers to it, set the did field to the did of some (existing) ’default’ department.

For every Works tuple that refers to it, set the did field to null.

1. CREATE TABLE Emp ( eid INTEGER, ename CHAR(10), age INTEGER, salary REAL, PRIMARY KEY (eid) ) CREATE TABLE Works ( eid INTEGER, did INTEGER, pcttime INTEGER, PRIMARY KEY (eid, did), FOREIGN KEY (did) REFERENCES Dept, FOREIGN KEY (eid) REFERENCES Emp, ON DELETE CASCADE) CREATE TABLE Dept ( did INTEGER, budget REAL, managerid INTEGER , PRIMARY KEY (did), FOREIGN KEY (managerid) REFERENCES Emp, ON DELETE SET NULL)
2. CREATE TABLE Dept ( did INTEGER, budget REAL, managerid INTEGER NOT NULL , PRIMARY KEY (did), FOREIGN KEY (managerid) REFERENCES Emp)
3. INSERT INTO Emp (eid, ename, age, salary) VALUES (101, ’John Doe’, 32, 15000)
4. UPDATE Emp E SET E.salary = E.salary \* 1.10
5. DELETE FROM Dept D WHERE D.dname = ’Toy’

The did field in the Works relation is a foreign key and references the Dept relation. This is the referential integrity constraint chosen. By adding the action ON DELETE CASCADE to this, when a department record is deleted, the Works record associated with that Dept is also deleted. The query works as follows: The Dept relation is searched for a record with name = ‘Toy’ and that record is deleted. The did field of that record is then used to look in the Works relation for records with a matching did value. All such records are then deleted from the Works relation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **sid** | **name** | **login** | **age** | **gpa** |
| 53831 | Madayan | madayan@music | 11 | 1.8 |
| 53832 | Guldu | guldu@music | 12 | 2.0 |

**Specify the foreign keys for this schema. [1M]**

**Populate the relations with a few sample tuples. [1M]**

**Example of an insertion in the SALE and SALESPERSON relations that violates the referential integrity constraints. [1M]**

**Example of an insertion in the SALE and SALESPERSON relations that does not violates the referential integrity constraints. [1.5M]**

**Q.8.Answer each of the following questions briefly. The questions are based on the followingrelational schema:**

**Emp(eid: integer, ename: string, age: integer, salary: real)**

**Works(eid: integer, did: integer, pcttime: integer)**

**Dept(did: integer, dname: string, budget: real, managerid: integer)**

**1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple?**

**2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints.**

**3. Define the Dept relation in SQL so that every department is guaranteed to have a manager.**

**4. Write an SQL statement to add John Doe as an employee with eid = 101, age = 32 and salary = 15,000.**

**5. Write an SQL statement to give every employee a 10 percent raise.**

**6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.**

1. Consider the following example. It is natural to require that the did field of Works should be a foreign key, and refer to Dept.

CREATE TABLE Works ( eid INTEGER NOT NULL , did INTEGER NOT NULL , pcttime INTEGER,

PRIMARY KEY (eid, did),

UNIQUE (eid),

FOREIGN KEY (did) REFERENCES Dept )

When a user attempts to delete a Dept tuple, There are four options:

Also delete all Works tuples that refer to it.

Disallow the deletion of the Dept tuple if some Works tuple refers to it.

For every Works tuple that refers to it, set the did field to the did of some

(existing) ’default’ department

For every Works tuple that refers to it, set the did field to null.

2. CREATE TABLE Emp( eid INTEGER,

ename CHAR(10),

age INTEGER,

salary REAL,

PRIMARY KEY (eid) )

CREATE TABLE Works ( eid INTEGER,

did INTEGER,

pcttime INTEGER,

PRIMARY KEY (eid, did),

FOREIGN KEY (did) REFERENCES Dept,

FOREIGN KEY (eid) REFERENCES Emp,

ON DELETE CASCADE)

CREATE TABLE Dept( did INTEGER,

budget REAL,

managerid INTEGER ,

PRIMARY KEY (did),

FOREIGN KEY (managerid) REFERENCES Emp,

ON DELETE SET NULL);

3. CREATE TABLE Dept ( did INTEGER, budget REAL, managerid INTEGER NOT NULL , PRIMARY KEY (did), FOREIGN KEY (managerid) REFERENCES Emp)

4. INSERT INTO Emp (eid, ename, age, salary) VALUES (101, ’John Doe’, 32, 15000);

5. UPDATE Emp E The Relational Model 27 SET E.salary = E.salary \* 1.10;

6. DELETE FROM Dept D WHERE D.dname = ’Toy’

**1. Give an example of a foreign key constraint that involves the Dept relation. What are the options for enforcing this constraint when a user attempts to delete a Dept tuple? [1M]**

**2. Write the SQL statements required to create the preceding relations, including appropriate versions of all primary and foreign key integrity constraints. [1M]**

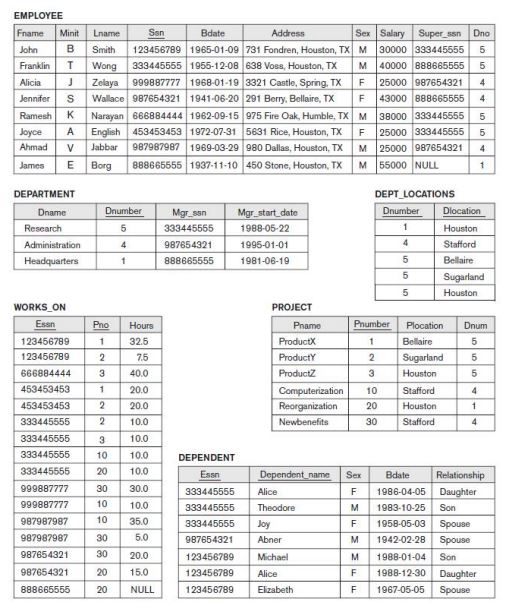
**3. Define the Dept relation in SQL so that every department is guaranteed to have a manager. [1M]**

**4. Write an SQL statement to add John Doe as an employee with eid = 101, age = 32 and salary = 15,000. [0.5M]**

**5. Write an SQL statement to give every employee a 10 percent raise. [0.5M]**

**6. Write an SQL statement to delete the Toy department. Given the referential integrity constraints you chose for this schema, explain what happens when this statement is executed.** **[0.5M]**

**Q.9. Suppose that each of the following Update operations is applied directly to the database state shown in the above figure. Discuss all integrity constraints violated by each operation, if any, and the different ways of enforcing these constraints.**



i. Insert < 'Robert', 'F', 'Scott', '943775543', '21-JUN-42', '2365 Newcastle Rd, Bellaire, TX', M, 58000, ‘888665555’, 1 > into EMPLOYEE..

ii. Delete the WORKS\_ON tuples with Essn = ‘333445555’.

iii. Modify the Mgr\_ssn and Mgr\_start\_date of the DEPARTMENT tuple with Dnumber = 5 to ‘123456789’ and ‘2007-10-01’, respectively **[2 M+2 M+4 M]**

i) This insertion satisfies all constraints, so it is acceptable

ii) No constraint violations.

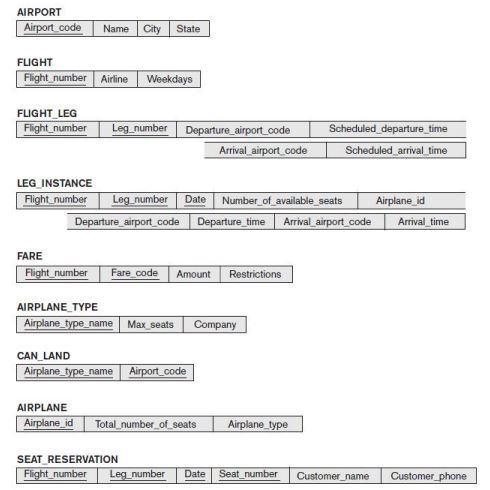
iii) No constraint violations.

**Q.10. Consider the AIRLINE relational database schema shown in the above schema which describes a database for airline flight information. Each FLIGHT is identified by a Flight\_number, and consists of one or more FLIGHT\_LEGs with Leg\_numbers 1, 2, 3, and so on. Each FLIGHT\_LEG has scheduled arrival and departure times, airports, and one or more LEG\_INSTANCEs— one for each Date on which the flight travels. FAREs are kept for each FLIGHT. For each FLIGHT\_LEG instance, SEAT\_RESERVATIONs are kept, as are the AIRPLANE used on the leg and the actual arrival and departure times and airports. An AIRPLANE is identified by an Airplane\_id and is of a particular AIRPLANE\_TYPE. CAN\_LAND relates AIRPLANE\_TYPEs to the AIRPORTs at which they can land. An AIRPORT is identified by an Airport\_code. Consider an update for the AIRLINE database to enter a reservation on a particular flight or flight leg on a given date.**

**a. Give the operations for this update.**

**b. What types of constraints would you expect to check?**

**c. Which of these constraints are key, entity integrity, and referential integrity constraints and which are not.**



**a)** One possible answer is given below:

INSERT <FNO,LNO,DT,SEAT\_NO,CUST\_NAME,CUST\_PHONE> into

SEAT\_RESERVATION; MODIFY the LEG\_INSTANCE tuple with the condition:

( FLIGHT\_NUMBER=FNO AND LEG\_NUMBER=LNO AND DATE=DT) by setting

NUMBER\_OF\_AVAILABLE\_SEATS = NUMBER\_OF\_AVAILABLE\_SEATS - 1;

These operations should be repeated for each LEG of the flight on which a reservation is made. This assumes that the reservation has only one seat. More complex operations will be needed for a more realistic reservation that may reserve several seats at once.

**b)** We would check that NUMBER\_OF\_AVAILABLE\_SEATS on each LEG\_INSTANCE of the flight is greater than 1 before doing any reservation (unless overbooking is permitted),

and that the SEAT\_NUMBER being reserved in SEAT\_RESERVATION is available.

**c)** The INSERT operation into SEAT\_RESERVATION will check all the key, entity integrity,and referential integrity constraints for the relation..The check that NUMBER\_OF\_AVAILABLE\_SEATS on each LEG\_INSTANCE of the flight is greater than 1 does not fall into any of the above types of constraints. (it is a general semantic integrity constraint).

**a. Give the operations for this update. [3M]**

**b. What types of constraints would you expect to check? [3M]**

**c. Which of these constraints are key, entity integrity, and referential integrity constraints and which are not. [2M]**

**Q.11. Design a relational database schema for a database application of your choice.**

**a. Declare your relations using the SQL DDL.**

**b. Specify a number of queries in SQL that are needed by your database application.**

**c. Based on your expected use of the database, choose some attributes that should have indexes specified on them.**

**d. Implement your database, if you have a DBMS that supports SQL.**

**CREATE TABLE STATEMENT**

DROP TABLE Enrollment;  
DROP TABLE offering;  
DROP TABLE Student;  
DROP TABLE Course;  
DROP TABLE Faculty;

-------------------- Student --------------------------------

CREATE TABLE Student (  
stdNo char(11) not null,  
stdFirstName varchar2(30) not null,  
stdLastName varchar2(30) not null,  
stdCity varchar2(30) not null,  
stdState char(2) not null,  
stdZip char(10) not null,  
stdMajor char(6),  
stdClass char(2),  
stdGPA decimal(3,2),  
CONSTRAINT StudentPk PRIMARY KEY (StdNo) );

-------------------- Course --------------------------------

CREATE TABLE Course(  
CourseNo   char(6) not null,  
crsDesc       varchar2(50) not null,  
CrsUnits   integer,  
CONSTRAINT CoursePK PRIMARY KEY (CourseNo) );

-------------------- Faculty --------------------------------

CREATE TABLE Faculty(  
FacNo           char(11) not null,  
FacFirstName   varchar2(30) not null,  
FacLastName       varchar2(30) not null,  
FacCity           varchar2(30) not null,  
FacState       char(2) not null,  
FacZipCode       char(10) not null,  
FacRank           char(4),  
FacHireDate       date,  
FacSalary       decimal(10,2),  
FacSupervisor   char(11),  
FacDept           char(6),  
CONSTRAINT FacultyPK PRIMARY KEY (FacNo),  
CONSTRAINT SupervisorFK FOREIGN KEY (FacSupervisor) REFERENCES Faculty );

-------------------- Offering --------------------------------

CREATE TABLE Offering(  
OfferNo INTEGER not null,  
CourseNo char(6) not null,  
OffTerm char(6) not null,  
OffYear INTEGER not null,  
OffLocation varchar2(30),  
OffTime varchar2(10),  
FacNo char(11),  
OffDays char(4),  
CONSTRAINT OfferingPK PRIMARY KEY (OfferNo),  
CONSTRAINT CourseFK FOREIGN KEY (CourseNo) REFERENCES Course,  
CONSTRAINT FacultyFK FOREIGN KEY (FacNo) REFERENCES Faculty );

-------------------- Enrollment --------------------------------

CREATE TABLE Enrollment (  
OfferNo       INTEGER not null,  
StdNo       char(11) not null,  
EnrGrade   decimal(3,2),  
CONSTRAINT EnrollmentPK PRIMARY KEY (OfferNo, StdNo),  
CONSTRAINT OfferingFK FOREIGN KEY (OfferNo) REFERENCES Offering  
ON DELETE CASCADE,  
CONSTRAINT StudentFK FOREIGN KEY (StdNo) REFERENCES Student ON DELETE CASCADE );

**INSERT STATEMENT (SOME MAY BE NOT BE INSERTED FOR SHOWING YOU CONSTRAINT )**

INSERT INTO student  
   (stdNo, stdFirstName, stdLastName, stdCity,  
   stdState, stdMajor, stdClass, stdGPA, stdZip)  
   VALUES ('123-45-6789','HOMER','WELLS','SEATTLE','WA','IS','FR',3.00,'98121-1111');

INSERT INTO student  
   (stdNo, stdFirstName, stdLastName, stdCity,  
   stdState, stdMajor, stdClass, stdGPA, stdZip)  
   VALUES ('124-56-7890','BOB','NORBERT','BOTHELL','WA','FIN','JR',2.70,'98011-2121');

INSERT INTO student  
   (stdNo, stdFirstName, stdLastName, stdCity,  
   stdState, stdMajor, stdClass, stdGPA, stdZip)  
   VALUES ('234-56-7890','CANDY','KENDALL','TACOMA','WA','ACCT','JR',3.50,'99042-3321');

INSERT INTO student  
   (stdNo, stdFirstName, stdLastName, stdCity,  
   stdState, stdMajor, stdClass, stdGPA, stdZip)  
   VALUES ('345-67-8901','WALLY','KENDALL','SEATTLE','WA','IS','SR',2.80,'98123-1141');

INSERT INTO student  
   (stdNo, stdFirstName, stdLastName, stdCity,  
   stdState, stdMajor,...

**a. Declare your relations using the SQL DDL. [2M]**

**b. Specify a number of queries in SQL that are needed by your database application.[2M]**

**c. Based on your expected use of the database, choose some attributes that should have indexes specified on them. [4M]**

**d. Implement your database, if you have a DBMS that supports SQL. [4.5M]**

**Q.12. Consider the following MAILORDER relational schema describing the data for a mail order company.**

**PARTS(Pno, Pname, Qoh, Price, Olevel)**

**CUSTOMERS(Cno, Cname, Street, Zip, Phone)**

**EMPLOYEES(Eno, Ename, Zip, Hdate)**

**ZIP\_CODES(Zip, City)**

**ORDERS(Ono, Cno, Eno, Received, Shipped)**

**ODETAILS(Ono, Pno, Qty)**

**Qoh stands for quantity on hand: the other attribute names are self-explanatory. Specify and execute the following queries using the RA interpreter on the MAILORDER database schema.**

**a. Retrieve the names of parts that cost less than $20.00.**

**b. Retrieve the names and cities of employees who have taken orders for parts costing more than $50.00. c. C.Retrieve the pairs of customer number values of customers who live in the same ZIP Code**

**d. Retrieve the names of customers who have ordered parts from employees living in Wichita**

**e. Retrieve the names of customers who have ordered parts costing less than $20.00.**

**f. Retrieve the names of customers who have not placed an order.**

**g. Retrieve the names of customers who have placed exactly two orders.**

1. Select Pname from PARTS where Price < 20.
2. select e.ename,z.city from employees e, zipcodes z, orders o, odetails d, parts p where e.zip = z.zip and e.eno = o.eno and o.ono = d.ono and d.pno = p.pno and p.price> 50.00;
3. select c1.cno, c2.cno from customers c1, customers c2 where c1.zip = c2.zip and c1.cno < c2.cno;
4. select cname from customers where not exists (select \* from orders o, employees e, zipcodes z where o.cno = customers.cno and o.eno = e.eno and e.zip = z.zip and z.city<> 'Wichita');
5. select cname from customers c where not exists (select \* from parts p where p.price< 20.00 and not exists (select \* from orders o, odetails d where o.ono = d.ono and o.cno = c.cno and o.pno = p.pno));
6. select cname from customers where not exists (select \* from orders where orders.cno = customers.cno);
7. select cname from customers where exists (select \* from orders o1, orders o2 where o1.cno = customers.cno and o2.cno = customers.cno and o1.ono <> o2.ono) and not exists ( select \* from orders o1, orders o2, orders o3 where o1.cno = customers.cno and o2.cno = customers.cno and o3.cno = customers.cno and o1.ono <> o2.ono and o2.ono <> o3.ono and o1.ono <> o3.ono);

**a. Retrieve the names of parts that cost less than $20.00. [1M]**

**b. Retrieve the names and cities of employees who have taken orders for parts costing more than $50.00. [1.5M]**

**c. Retrieve the pairs of customer number values of customers who live in the same ZIP Code. [2M]**

**d. Retrieve the names of customers who have ordered parts from employees living in Wichita. [2M]**

**e. Retrieve the names of customers who have ordered parts costing less than $20.00. [2M]**

**f. Retrieve the names of customers who have not placed an order. [2M]**

**g. Retrieve the names of customers who have placed exactly two orders. [2M]**