**Department of Computer Science and Engineering**

**Semester VI Database Management Systems Laboratory exercises**

**(December 2019 to April 2020)**

**Exercises based on Database Implementation**

**(Use C/C++/Java/Python)**

**Exercise –I**

i) Create a data file to store records of the students (fields: rollno, name, branch,age). (ii) Sort the records of the file based on the rollno of the students.

(iii) Perform external sorting procedure (based on the roll number) on two data files which store records of the students and store the result in to the third data file.

(iv) Store student records (fields: rollno,name,branch,age) in a data file and perform linear search in the data file by reading *rollno* as input and then display the student details and display the time required to do this operation.

**Exercise 2**

Store student records (fields: rollno,name,branch,age) in a data file and build an index file by considering the *rollno* as the key.

1. Perform linear search in the index file by reading *rollno* as input and then display the student details by reading from the data file and display the time required to do this operation.
2. Perform binary search in the index file (by sorting the index file based on the *rollno*) by reading *rollno* as input and then display the student details by reading from the data file and display the time required to do this operation.

**Exercise –3**

Store student records (fields: rollno,name,branch,age) in a data file and build an index file by using binary search tree ( *rollno* is usedas the key).

1. Perform search in the index file by reading *rollno* as input and then display the student details by reading from the data file and display the time required to do this operation.
2. Add and delete the student records from the data file and then perform corresponding modifications in the index file.

**Exercise –4**

Store student records (fields: rollno,name,branch,age) in a data file and build an index file by using hash table (*rollno* is usedas the key here).

1. Perform search in the index file by reading *rollno* as input and then display the student details by reading from the data file and display the time required to do this operation.
2. Add and delete the student records from the data file and then perform corresponding modifications in the index file.

**Exercise – 5**

Store student records (fields: rollno,name,branch,age) in a data file and build an index file by using B+ tree (*rollno* is usedas the key here).

1. Perform search in the index file by reading *rollno* as input and then display the student details by reading from the data file and display the time required to do this operation.
2. Add and delete the student records from the data file and then perform corresponding modifications in the index file.

**Exercise-6**

Develop module(s) for creating tables by fixing the number of columns (fields), individual column names and their data types and store the schema in the catalog (data dictionary). Consider one column as the primary key and create index file for the same. Also, develop the module for dropping a table along with the index file and for modifying the table by adding/dropping columns.

**Exercise –7**

1. Develop module(s) for inserting rows into the selected table and for deleting the rows of the selected table by including the facility to specify conditions for deletion.
2. Develop module(s) for reading the data present in the table and also include the facility to specify the conditions for selecting the rows from the table. Use the index file (if needed) for the quick access of data. Try to have index files based on various columns (fields) of the table. And also develop an indexing method using which it is possible to execute the range queries in a faster manner.

**Exercise – 8**

Develop a prototype DBMS by combining all the modules that you have developed for Exercises I to VII by including an excellent GUI for manipulating the data present in the tables.

SQL-Based Exercises

Exercise 9

Consider the following relations.

Suppliers( *sid:* integer, *sname:* string, *address:* string)

Parts(*pid:* integer, *pname:* string, *color:* string)

Catalog*( sid:* integer, *pid:* integer, *cost:* real)

The key fields are underlined, and the domain of each field is listed after the field name.

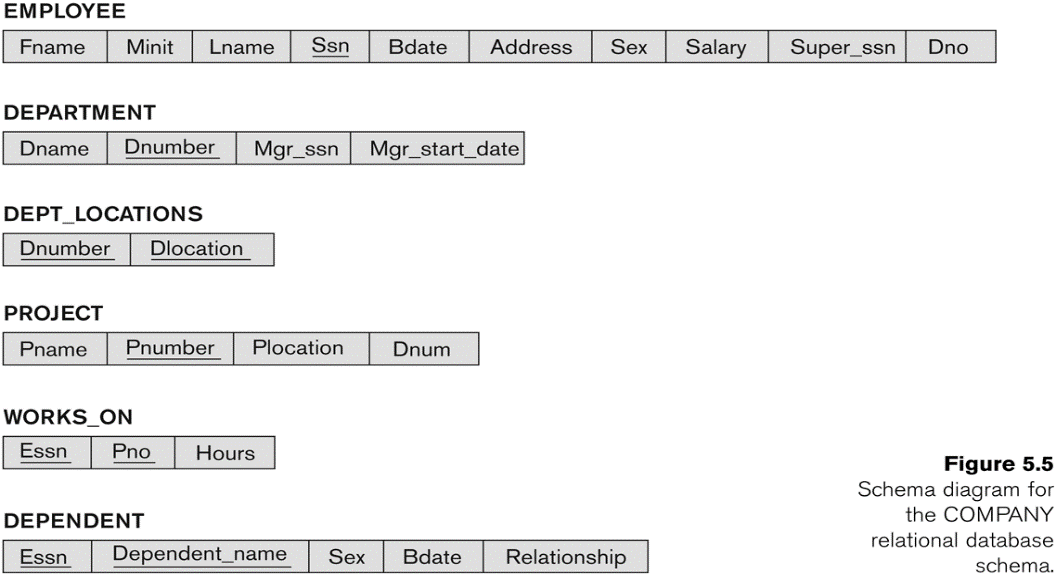
Therefore *sid* is the key for Suppliers, *pid* is the key for Parts, and *sid* and *pid* together form

the key for Catalog. The Catalog relation lists the prices charged for parts supplied by Suppliers.

Write SQL statements for the following.

1. Find the *names* of suppliers who supply some **red color** part.
2. Find the *sids* of suppliers who supply some **red color** part and having office located at ‘Chennai’
3. Find the average cost of **red color** parts supplied by various suppliers.
4. Find the names of the supplier who is supplying most number of parts.
5. Find the *sids* of suppliers who supply every part.
6. Find the *sids* of suppliers who supply every red color part.
7. List the number of suppliers for each color of part.
8. Find the supplier who supplies the **red color** part at a cheaper rate.
9. For each color part, display the details of the suppliers who supply that part at a cheaper rate.
10. Display the names of the suppliers along with the number of parts supplied by them.
11. Find the details of the supplier who supplies the costliest part.
12. Display the names of the suppliers who are selling at least two parts.

1. A) Consider the COMPANY database schema shown in the figure.



1. Create a view that has department name, manager name and manager salary for every department.
2. Create a view that has project name, controlling depart name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it.
3. Create an updateable view for the relation DEPARTMENT

B) Create a materialized view for finding average salary of employees, average salary of managers, average salary for each department and department(s) which spend more money on salary for the employees.

C) Assume that Dno of EMPLOYEE relation has got NOT NULL constraint. Write a transaction which inserts tuples in to the relations EMPLOYEE and DEPARTMENT without affecting integrity constraints specified in the schema.

1. A) Consider the following relations:

*instructor*(*ID*, *name*, *dept\_name*, *salary*)

*section*(*course\_id*, *sec\_ id*, *semester*, *year*, *building*, *room\_ number*, *time\_slot\_id*)

*teaches*(*ID*, *course\_id*, *sec\_id*, *semester*, *year*)

***Write assertions for the following:***

1. An instructor cannot teach in two different classrooms in a semester in the same slot
2. An instructor cannot teach more than one course for the same semester

B) Consider the following relations.

*product(maker, model, type)*

*pc(model, speed, ram, hd, price )*

*laptop(model, speed, ram, hd, screen , price )*

*printer(model, color, type, price )*

***Write triggers for the following:***

(a) When updating the price of a PC, check that there is no lower priced PC with the same speed.

(b) When inserting a new printer, check that the model number exists in product.

(c) When making any modification to the Laptop relation, check that the average price of laptops for each manufacturer is at least Rs 1500.

C) Consider the following relations.

emp(eno,ename,eage, salary,departno,supereno), dep(depno,depname,depage,eno), depart(departno,departname,location)

Write stored procedures

1. to find the average salary of employees who have got more than two dependents
2. to find the names of employees (age is greater than 50) and their dependents (average age is less than 10).
3. Write java programs (using JDBC)
   1. to create the following relations emp(eno,ename,eage, salary,departno,supereno), dep(depno,depname,depage,eno), depart(departno,departname,location) and insert at least 20 tuples for each relation.
   2. (i) to find average age of employees department wise (ii) to list department(s) (location wise) which pay less salary to the employees.