Leading University

Department of Computer Science and Engineering

Assignment

Course Title: ***Database Management System***

Course Code: ***CSE-2319***

Submitted To

Shafkat Kibria

Assistant Professor & Head(Acting)

Department of Computer Science and Engineering

Leading University

Submitted By

Nader Nihal Neep

ID-2012020022

Section: A

Solutions

Question#1

a) **DBMS**: A database management system or DBMS is a collection of interrelated data and a set of programs that allow users to access and modify these data. Database systems are used to manage collections of data that are: Highly valuable, relatively large and accessed by multiple users and applications, often at the same time.

b)**DDL**: The SQL data-definition language (DDL) allows the specification of information about relations, including: The schema for each relation, the domain of values associated with each attribute, Integrity constraints.

c) **DML**: DML or Data Manipulation Language is a language for accessing and updating the data organized by the appropriate data model. DML is also known as query language.

d)**XML**: XML or Extensible Markup Language is originally intended as a document markup language not a database language. It has the ability to specify new tags, and to create nested tag structures made XML a great way to exchange data, not just Documents. XML has become the basis for all new generation data interchange formats. A wide variety of tools is available for parsing, browsing and querying XML documents/data.

e) **Storage Manager**: A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.The storage manager is responsible to the following tasks: Interaction with the OS file manager, efficient storing, retrieving and updating of data the storage manager components include: authorization and integrity manager, transaction manager, file manager, buffer manager.

f)**Domain Types of SQL**:

- char(n). Fixed length character string, with user-specified length n.

• varchar(n). Variable length character strings, with user-specified maximum length n.

• int. Integer (a finite subset of the integers that is machine-dependent).

• smallint. Small integer (a machine-dependent subset of the integer domain type).

• numeric(p,n). Fixed point number, with user-specified precision of p digits, with n digits to the right of decimal point. • real, double precision. Floating point and double-precision floating point numbers, with machine-dependent precision.

• float(n). Floating point number, with user-specified precision of at least n digits.

g) **Nonprocedural language**: A computer language that require a user to specify what data are needed without specifying how to get those data. It is also known as 'Declarative Language'.

h)**Data Model**: A collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.

i) **Relation Schema & Instance**: Relation schema defines the design and structure of the relation like it consists of the relation name, set of attributes or column names.Every attribute would have an associated domain.

Instance is the actual content of the database at a particular point in time.

j) **Storage Manager**: A program module that provides the interface between the low-level data stored in the database and the application programs and queries submitted to the system.The storage manager is responsible to the following tasks: Interaction with the OS file manager, efficient storing, retrieving and updating of data the storage manager components include: authorization and integrity manager, transaction manager, file manager, buffer manager.

Question#2

Difficulty in accessing data: Need to write a new program to carry out each new task.

- Data isolation: Multiple files and formats.
- Integrity problems: Integrity constraints (e.g., account balance > 0) become "buried" in program code rather than being stated explicitly. Hard to add new constraints or change existing ones.
- Atomicity of updates: Failures may leave database in an inconsistent state with partial updates carried out.
- Concurrent access by multiple users: Uncontrolled concurrent accesses can lead to inconsistencies.
- Security problems: Hard to provide user access to some, but not all, data.

b. The levels of abstraction in database system are given below:

- Physical level: describes how a record is stored.
- Logical level: describes data stored in database, and the relationships among the data.
- View level: application programs hide details of data types. Views can also hide information (such as an employee's salary) for security purposes.

Question#3

a) I. $\sigma_{B=D}(s)$

Result:

s.B	s.C	s.D
3	21	3
2	4	2

II. $\Pi_{C,D}(\sigma_{D>2}(s))$

Result:

s.C	s.D
2	3
21	3
21	20

III. $r \times s$

Result:

r.A	r.B	s.B	s.C	s.D
11	2	1	2	3
11	2	3	21	3
11	2	2	4	2
11	2	5	21	20
21	12	1	2	3
21	12	3	21	3
21	12	2	4	2

21	12	5	21	20
11	5	1	2	3
11	5	3	21	3
11	5	2	4	2
11	5	5	21	20

IV. r ⋈s

Result:

r.A	r.B	s.C	s.D
11	2	4	2
11	5	21	20

V. r ⋊⋉s

Result:

r.A	r.B	s.C	s.D
11	2	4	2
21	12	null	null
11	5	21	20

Question#4

a. The SQL for computing Department-wise average salary from the Instructor table :

```
select dept_name, avg (salary) avg_salary
                from instructor
group by dept_name;
```

b. The result using the instructor table:

NAME
Einstein
Brandt
Wu
Gold
Kim
Singh
Katz

c. The result of the SQL using the course and prereq table:

course_id	title	dept_name	credits	prereq_id
BIO-301	Genetics	Biology	4	BIO-101
CS-190	Game Design	Comp.Sci.	4	CS-101
CS-347	null	null	null	CS-101

Question#5

a. The correct SQL is written below:

```
insert into course (Course_id, Course_Name, Dept )
values ('CS-437', 'Database Systems', 'Comp. Sci.');

update instructor
    set salary = salary * 1.03
where salary > 100000;
```

b. I. The output for student natural left outer join Takes is given below:

STUDENT_ID	NAME	ADDRESS	TOTAL_CREDIT_COMPLETE	SESSION_ID	COURSE_ID
0801CSE00505	Y	YYY	41	2	CSE101
0801CSE00508	W	WWW	47	null	null
0801CSE00503	X	XXX	21	1	CSE201
0801CSE00507	Z	ZZZ	52	1	CSE304
0801CSE00505	Y	YYY	41	2	CSE205
0801CSE00507	Z	ZZZ	52	1	CSE201
0801CSE00507	Z	ZZZ	52	2	CSE209
0801CSE00505	Y	YYY	41	1	CSE209

II. The output for: select *

from student, Takes;

is given below:

Student.STUDENT_ID	NAME	ADDRESS	TOTAL_CREDIT_COMPLETE	SESSION_ID	COURSE_ID	Takes.STUDENT_ID
0801CSE00503	X	XXX	21	1	CSE201	0801CSE00503
0801CSE00503	X	XXX	21	1	CSE304	0801CSE00507
0801CSE00503	X	XXX	21	2	CSE205	0801CSE00505
0801CSE00503	X	XXX	21	1	CSE201	0801CSE00507
0801CSE00503	X	XXX	21	2	CSE209	0801CSE00507
0801CSE00503	X	XXX	21	1	CSE209	0801CSE00505
0801CSE00503	X	XXX	21	2	CSE101	0801CSE00505
0801CSE00505	Y	YYY	41	1	CSE201	0801CSE00503
0801CSE00505	Y	YYY	41	1	CSE304	0801CSE00507
0801CSE00505	Y	YYY	41	2	CSE205	0801CSE00505
0801CSE00505	Y	YYY	41	1	CSE201	0801CSE00507
0801CSE00505	Y	YYY	41	2	CSE209	0801CSE00507
0801CSE00505	Y	YYY	41	1	CSE209	0801CSE00505
0801CSE00505	Y	YYY	41	2	CSE101	0801CSE00505
0801CSE00507	Z	ZZZ	52	1	CSE201	0801CSE00503
0801CSE00507	Z	ZZZ	52	1	CSE304	0801CSE00507
0801CSE00507	Z	ZZZ	52	2	CSE205	0801CSE00505
0801CSE00507	Z	ZZZ	52	1	CSE201	0801CSE00507
0801CSE00507	Z	ZZZ	52	2	CSE209	0801CSE00507
0801CSE00507	Z	ZZZ	52	1	CSE209	0801CSE00505
0801CSE00507	Z	ZZZ	52	2	CSE101	0801CSE00505
0801CSE00508	W	WWW	47	1	CSE201	0801CSE00503
0801CSE00508	W	WWW	47	1	CSE304	0801CSE00507
0801CSE00508	W	WWW	47	2	CSE205	0801CSE00505
0801CSE00508	W	WWW	47	1	CSE201	0801CSE00507
0801CSE00508	W	WWW	47	2	CSE209	0801CSE00507
0801CSE00508	W	WWW	47	1	CSE209	0801CSE00505
0801CSE00508	W	WWW	47	2	CSE101	0801CSE00505

III. The output for student natural full outer join Takes is given below:

STUDENT_ID	NAME	ADDRESS	TOTAL_CREDIT_COMPLETE	SESSION_ID	COURSE_ID
0801CSE00503	X	XXX	21	1	CSE201
0801CSE00507	Z	ZZZ	52	1	CSE304
0801CSE00505	Y	YYY	41	2	CSE205
0801CSE00507	Z	ZZZ	52	1	CSE201
0801CSE00507	Z	ZZZ	52	2	CSE209
0801CSE00505	Y	YYY	41	1	CSE209
0801CSE00505	Y	YYY	41	2	CSE101
0801CSE00508	W	WWW	47	null	null

c. A select SQL to find the Student, who are not registered in the 1st Session:

```
select *
from student natural join takes
where Session_id !='1';
```

Question#6

a. **Referential Integrity**: It ensures that a value that appears in one relation for a given set of attributes also appears for a certain set of attributes in another relation.
Example: If “Biology” is a department name appearing in one of the tuples in the instructor relation, then there exists a tuple in the department relation for “Biology”.

Let A be a set of attributes. Let R and S be two relations that contain attributes A and where A is the primary key of S. A is said to be a foreign key of R if for any values of A appearing in R these values also appear in S.

Now an example using student and takes table is given below:

```
create table student(

student_id                  varchar(20),
name                        varchar(20),
address                     varchar(20),
Total_credit_complete       number(20),
primary key(student_id));

insert into student values ('0801CSE00503', 'X','XXX',21);
insert into student values ('0801CSE00505', 'Y','YYY',41);
insert into student values ('0801CSE00507', 'Z','ZZZ',52);
insert into student values ('0801CSE00508', 'W','WWW',47);

create table takes(

session_id     number(20),
course_id      varchar(20),
student_id varchar(20),
foreign key(student_id) references  student);

insert into takes values (1,'CSE201', '0801CSE00503');
insert into takes values (1,'CSE304', '0801CSE00507');
insert into takes values (2,'CSE205', '0801CSE00505');
insert into takes values (1,'CSE201', '0801CSE00507');
insert into takes values (2,'CSE209', '0801CSE00507');
```

```
insert into takes values (1,'CSE209', '0801CSE00505');
insert into takes values (2,'CSE101', '0801CSE00505');
```

b. An example of Primary Key Constraint using the table course is given below:

```
create table course(

course_id              varchar(20),
course_name          varchar(20),
credit                   number(20),
primary key(course_id));

insert into course values ('CSE201', 'Data Structure',4);

insert into course values ('CSE304', 'Database system',4);

insert into course values ('CSE101', 'Essential computing',3);

insert into course values ('CSE209', 'Digital Logic Design',3);
```

Question#7

Creation of tables student, takes and course

```
create table student(

    student_id                varchar(20),
```

```
name                    varchar(20),
address                 varchar(20),
Total_credit_complete   number(20),
primary key(student_id));

insert into student values ('0801CSE00503', 'X','XXX',21);
insert into student values ('0801CSE00505', 'Y','YYY',41);
insert into student values ('0801CSE00507', 'Z','ZZZ',52);
insert into student values ('0801CSE00508', 'W','WWW',47);


create table takes(

session_id    number(20),
course_id     varchar(20),
student_id varchar(20),
foreign key(student_id) references  student);

insert into takes values (1,'CSE201', '0801CSE00503');
insert into takes values (1,'CSE304', '0801CSE00507');
insert into takes values (2,'CSE205', '0801CSE00505');
insert into takes values (1,'CSE201', '0801CSE00507');
insert into takes values (2,'CSE209', '0801CSE00507');
insert into takes values (1,'CSE209', '0801CSE00505');
insert into takes values (2,'CSE101', '0801CSE00505');


create table course(

course_id           varchar(20),
course_name         varchar(20),
credit              number(20),
```

```
primary key(course_id));

insert into course values ('CSE201', 'Data Structure',4);

insert into course values ('CSE304', 'Database system',4);

insert into course values ('CSE101', 'Essential computing',3);

insert into course values ('CSE209', 'Digital Logic Design',3);
```

Creation of view

```
create view student_info as
select name
from student
where student_id in(select student_id
from takes
where course_id in(select course_id
from course
where course_name='Digital Logic Design'));
```