School : Computer Science
Institution : University of Windsor
Term : Fall 2021
Course : Comp-3150 (03-60-315-1) : Database Management Systems
Instructor : Dr. C. I. Ezeife
Assignment #2 Solution : Total : 50 marks
Handed Out: Thurs. Sep. 30, 2021; Due Thurs. Oct. 28, 2021

Objective of Assignment: To test on knowledge and design of relational model constraints, relational database schemas, functional dependencies and normalization of relational databases.

Scope: Assignment covers materials from Chapters 5 and 14 of book discussed in class.

Electronic Assignment Submission: Done through http://blackboard.uwindsor.ca

Marking Scheme : The mark for each of the questions is indicated beside each question.

Academic Integrity Statement : Remember to submit only work that is yours and include the following confidentiality agreement and statement at the beginning of your assignment.

CONFIDENTIALITY AGREEMENT & STATEMENT OF HONESTY

I confirm that I will keep the content of this assignment/examination confidential.
I confirm that I have not received any unauthorized assistance in preparing for or doing this assignment/examination. I confirm knowing that a mark of 0 may be assigned for copied work.

_____ _____
Student Signature Student Name (please print)

_____ _____
Student I.D. Number Date

Marking Scheme : The mark for each question and sub question is shown with the question below. Place your solutions in tables where possible.

For office Use only

Question	Mark
1	/20
2	/10
3	/10
4	/10
Total	/50

CHAPTER 5: THE RELATIONAL DATA MODEL AND RELATIONAL DATABASE CONSTRAINTS

1. (total marks 20) Given the same simple Person-Vaccinatedin-Centre database schema that contains three files described as follows, answer the following questions with regards to this database.

(Total for que 1 is 10 marks)

Person (Ssn: integer, Name: string, Age: integer, jobtype: string)
Vaccinatedin (Ssn: integer, Cntid: integer, vacdate:date, dose: integer, time: real)
Centre (Cntid: integer, Cntname: string, city: string, budget: real, managerid: integer)

Note: Ssn, Name, Age, jobtype are the social security number, name, age and job type respectively. Also, Cntid, vacdate, dose and time represent centre id, vaccination date (e.g., in dd-mon-yy), which dose (1 or 2) and vaccination time (e.g., 0.00 is 12.00am and 12.00 is 12.00pm at noon). The rest of the attributes Cntname, city, budget and managerid are the centre name (eg. WFCU, St Clair, DownTown, Other), city, budget for running the centre (eg. $80,000) and managerid respectively. A manager is a Person. A manager (e.g., with managerid same as an SSN) is a person who manages the vaccination centre.

Assume that an update operation (a general term in this chapter, for an insert, a modify or a delete operation) is to be made to this database to enter information about a new person not already in the database but who has just been vaccinated in a centre. Answer the following questions on what specific relations, attributes and operations (eg. insert, modify, delete) that need to be done for this update to be implemented in the entire database. This is not SQL query yet.

Provide your answers both in descriptive sentence and using the formal (not SQL) database operations of INSERT, MODIFY, DELETE as used in Chapter 5 of book with specific attributes and relations when possible. An example formal insert of a peson record into the Member table is:
INSERT < Ssn, Name, Age, jobtype> into Person // for new Person record
And an example descriptive sentence is:
i). do an insert operation for a new person record into the Person table.

(a) Give the set of needed insert, modify or delete operations for this update operation scenario described above. 5 marks
(b) What types of integrity constraints (explain using attributes, eg, SSN of relevant files) would you expect to check for this update to be done? 5 marks
(c) Which of these integrity constraints are key, entity integrity, and referential (foreign key) integrity constraints and which are not? 5 marks
(d) Specify all the referential integrity (foreign key) constraints on this database in the format Referring_Relation.Attribute --> Referred_Relation.Attribute.

5 marks
(Total for que 1 is 20 marks)

Solution:

Question	**Answers**
a. Give the operations for this update. 5 marks	One possible set of operations for the given update is the following: i. Insert operation into Person for the new Person record ii. Insert operation into Vaccinatedin for for each new vaccination of a new or existing person. These operations can be specified in more formal English (not SQL) operations as follows: INSERT < Ssn, Name, Age, jobtype> into Person // for new Person record INSERT <Ssn, Cntid, vacdate, dose, time> into Vaccinatedin // for the vaccination record of the new person
b. What types of integrity constraints would you expect to check? (explain using attributes, eg, Ssn of relevant files) 5 marks	We would check that with the first INSERT operation that the SSN assigned to the new person does not already exist (key constraint). With the second INSERT operation, we would check that for the new Vaccinatedin tuple, that the SSN in Person already exists in Person file and the newly inserted Vaccinatedin.Cntid already exists in the Centre file (referential integrity constraints).
c. Which of these integrity constraints are key, entity integrity, and referential integrity constraints and which are not? 5 marks	The INSERT operations into PERSON and VACCINATEDIN will check all the key, entity integrity, and referential integrity constraints for those relations.
d. Specify all the referential integrity constraints on this database. 5 marks	We will write a referential integrity constraint as R.A --> S.A (or R.(X) --> T.A) whenever attribute A (or the set of attributes X) of relation R form a foreign key that references the primary key of relation S (or T). VACCINATEDIN.SSN → PERSON.SSN VACCINATEDIN.CNTID → PERSON.CNTID

2. (total marks 10) Using your own Person-Vaccinatedin-Centre database instance from assignment 1, login to the SQL query processor on our cs server, called Oracle Sqlplus to create the three database tables and insert the tuples in your database state with the

following sequence of instructions. Note that this exercise is to get you beginning to connect to SQLplus while preparing to learn full SQL language syntax in Chapters 6 and 7. You will be given the instructions to use now. Show the result of this exercise through a Unix script file you will attach as a .txt file.

(Total for que 2 is 10 marks)

i. First connect to our cs.uwindsor.ca through either Bitvise SSH client or NoMachine.
ii. Then hand in a Unix script file to capture your Unix session when you connect to Sqlplus after your instructions for creating your database are working. You can create your Unix script file using the following sequence of instructions on a Unix terminal on our cs server. You need to transfer this script file to your personal computer using a file transfer protocol (eg. Bitvise SFTP or Filezilla) in order to attach it in your assignment submission.

```
>script   username_assn2que2.txt
>sqlplus <username>
>password

SQL>   CREATE TABLE PERSON(
SSN NUMBER(3) NOT NULL,
NAME    VARCHAR2(15),
AGE NUMBER(3),
JOBTYPE  VARCHAR2(15),
PRIMARY KEY(SSN));

SQL> CREATE TABLE CENTRE
 (
 CNTID  NUMBER(3) NOT NULL,
 CNTNAME VARCHAR2(15) NOT NULL,
 CITY    VARCHAR2(15),
 BUDGET  NUMBER(10,2),
 MANAGERID NUMBER(3),
 PRIMARY KEY (CNTID),
 FOREIGN KEY(MANAGERID) REFERENCES PERSON(SSN));

SQL> CREATE TABLE VACCINATEDIN(
SSN   NUMBER(3)  NOT NULL,
CNTID  NUMBER(3) NOT NULL,
VACDATE  DATE,
DOSE   NUMBER(3),
VACTIME NUMBER(6,2),
PRIMARY KEY(SSN, CNTID),
FOREIGN KEY(SSN) REFERENCES PERSON(SSN),
FOREIGN KEY(CNTID) REFERENCES CENTRE(CNTID));
```

```
SQL> -- A sample insert of a record into each of the 3 tables is given below
SQL> INSERT INTO PERSON VALUES (10, 'Jobe Bata', 65, 'Nurse');
SQL> INSERT INTO CENTRE VALUES(1, 'DownTn', 'Windsor', 6000000, 10);
SQL> INSERT INTO VACCINATEDIN VALUES (10, 1,  '02-apr-21', 1, 13.30);

SQL> COMMIT;
// Repeat similar INSERT instructions for all the data in all your tables
// starting with the entity tables first, eg, Person, Centre, before VACCINATEDIN.

SQL> select * from cat;   // to show all the objects in your catalogue

SQL> select * from Person;  // to show the contents of this table

SQL> exit                    //to exit sqlplus


>exit                          // to exit and create Unix script file to hand in
```

** More Hint: While in Sqlplus, if you want to delete data from your tables and drop them before issuing your instructions for creating your Unix script file for handing in, you can use the following instructions for each table to first delete the data from the table and then drop the table.

```
delete from VACCINATEDIN;
delete from PERSON;
delete from CENTRE;

commit;

drop table VACCINATEDIN cascade constraints;

drop table PERSON cascade constraints;

drop table CENTRE cascade constraints;

commit;
```
***

Also Note: you can start creating a script file only after you have created your tables correctly and inserted data in the tables. In that case, you cannot re-create existing tables. Then, you can just run the desc table (eg. Desc Person) command for each table to show the structure of each table before using (for example), the (select * from Person;) to show the tuples of each table or delete data and drop the tables as explained above so you can re-create the tables more correctly.

Solution: (10 marks)

An attached Unix script file showing execution of CREATE TABLE instructions and INSERT INTO tablename VALUES instructions with the few SELECT instructions to show contents of the catalogue and all tables (your database instance).

CHAPTER 14: Database Design Theory: Introduction to Normalization Using Functional and Multivalued Dependencies

3. (total marks 10) Consider the following relation:

Enrolled(Studid, Crsid, SName, Score, Lettergrade)

Assume that a student (Studid) may be enrolled in multiple courses (Crsid) and hence {Studid, Crsid } is the primary key.
Thus, the following functional dependency exists:

{Studid, Crsid } -> {SName, Score, Lettergrade}

Additional dependencies are:

Studid -> SName

Score -> Lettergrade

Based on the given primary key,

i. is this relation in 1NF, 2NF, or 3NF? Why or why not?
ii. If not in 2NF at least, normalize it completely into 2NF and 3NF? Provide your answers using functional dependencies (FDs).

(Total for que 3 is 10 marks)

Solution (i): (5 marks)

Answer:
Given the relation schema
Enrolled(Studid, Crsid, SName, Score, Lettergrade)

with the functional dependencies
{Studid, Crsid } -> {SName, Score, Lettergrade}
Studid -> SName
Score -> Lettergrade
(i) is this relation in 1NF, 2NF, or 3NF? Why or why not?

This relation satisfies 1NF but not 2NF because there is part of the primary key that determines a non-key attribute (Studid -> SName).

So the attribute (Sname) is not FFD on the primary key {Studid, Crsid } and the relation is not in 3NF because it is not in 2NF and there is transitive dependency between the primary key and some non-key attributes, e.g., Lettergrade (eg., {Studid, Crsid } -> Score;
and Score -> Lettergrade).

Solution (ii) (5 marks)

(ii) If not in 2NF at least, normalize it completely into 2NF and 3NF? Provide your answers using functional dependencies (FDs).

To normalize into 2NF and 3NF, we break the relations into relations that have only FDs that are FFD on the primary key for 2NF and also relations that have only FDs in each relation that are not transitively dependent on the primary key. The results are given below.

2NF: (keep only relations with FDs that are FFD on PK)
Enrolled1(Studid, Crsid, Score, Lettergrade)
Enrolled2(Studid, SName)

3NF: (Also remove the transitive dependencies)
Enrolled1(Studid, Crsid, Score)
Enrolled2(Studid, SName)
Enrolled3(Score, Lettergrade)

4. (total marks 10) What (i) update, (ii) delete and (iii) insertion anomalies occur in the DEPARTMENT_PROJECT relation obtained by doing a natural join of the two relations DEPARTMENT and PROJECT of Fig 14.2 on page 463 of book? Explain with examples using this database and the DEPARTMENT_PROJECT relation schema with state given below as Figures 4.1 and 4.2 below.

(Total for que

4 is 10 marks)
Note: 3 marks for correct discussion of each anomaly and 1 marK for attempt.

Figure 14.2 (book): Sample database state for a simplified COMPANY relation DB

EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

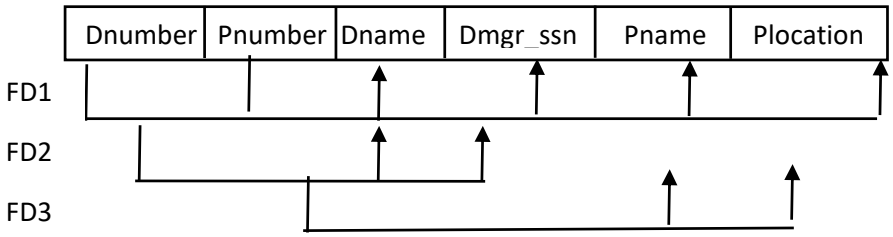
Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

****

Fig 4.1: DEPARTMENT_PROJECT DB schema suffering from update anomalies



***

Fig 4.2: A database state of the DEPARTMENT_PROJECT DATABASE derived from Fig 14.2

DNUMBER	PNUMBER	DNAME	DMGR_SSN	PNAME	PLOCATION
5	3	Research	333445555	ProductZ	Houston
5	10	Research	333445555	Computerize	Stafford
5	20	Research	333445555	Reorganize	Houston
5	30	Research	333445555	Nbenefits	Stafford
5	1	Research	333445555	ProductX	Bellair
5	2	Research	333445555	ProductY	Sugarland
4	3	Administration	987654321	ProductZ	Houston
4	10	Administration	987654321	Computerize	Stafford
4	20	Administration	987654321	Reorganize	Houston
4	30	Administration	987654321	Nbenefits	Stafford
4	1	Administration	987654321	ProductX	Bellair
4	2	Administration	987654321	ProductY	Sugarland
1	3	Headquarters	888665555	ProductZ	Houston
1	10	Headquarters	888665555	Computerize	Stafford
1	20	Headquarters	888665555	Reorganize	Houston
1	30	Headquarters	888665555	Nbenefits	Stafford
1	1	Headquarters	888665555	ProductX	Bellair
1	2	Headquarters	888665555	ProductY	Sugarland

18 rows selected.

Solution: (3 + 3 + 3 + 1 marks)

i. Update Anomalies: In DEPARTMENT_PROJECT, the partial dependencies {DNUMBER}->{DNAME} and {PNUMBER}->{PNAME,PLOCATION} can cause update anomalies. For example, all PROJECT records whose Pnumber is 2 (about 3 records) all have {PNAME, PLOCATION} as {ProductY, Sugarland}. We cannot update any of these records to something else or the database integrity is violated. If we fail to update some of the records to something else, the database is violated. We must remember these multiple column associations that indicate data redundancy and change them in all the associated row records or there is a violation and this is update anomaly.

ii. Delete Anomalies: : For example, if a PROJECT temporarily has no DEPARTMENTS working on it, its information (PNAME, PNUMBER, PLOCATION) will not be represented in the database when the last DEPARTMENT working on it is removed (this is deletion anomaly).

iii. Insertion anomaly: Inserting a new tuple relating an existing DEPARTMENT to an existing PROJECT requires checking both partial dependencies; for example, if a different value is entered for PLOCATION than those values in other tuples with the same value for PNUMBER, we get an update anomaly. Also, if a project is not yet assigned to a department, its record cannot be inserted into the database and this is insert anomaly.