**DSC 450 Winter Assignment-1**

**Name:** Muhammad Umair

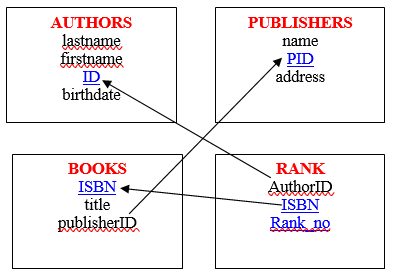
**Section:** In-class

**Part 1**

1. Create a relational schema with underlined (primary) keys and arrows connecting foreign keys and primary keys for a database containing the following information. If you have any difficulty drawing arrows, you can write foreign key information in a sentence instead.

* **Authors** have Last Name, Firstname, ID, and Birthdate (identified by ID)
* **Publishers** have Name, ID, address (identified by ID)
* **Books** have ISBN, Title, Publisher (each book has a unique publisher and can be identified by ISBN).
* Authors **Write** Books; since many authors can co-author a book, we need to know the rank of an author contributing to a book, stored in this table (i.e. a number 1, 2, 3; for single author books, this number is 1).

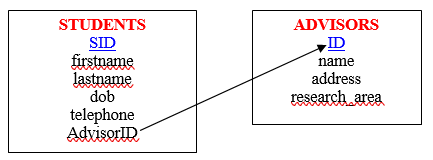
NOTE: Part 2 has some sample data which may be helpful.



* Primary Key for RANK: Combination of ISBN and Rank\_no

1. Create a relational schema for students and student advisors

* **Students** have First Name, Last Name, DOB, Telephone and a reference to their advisor
* **Advisors** have ID, Name, Address, Research Area



**Part 2**

1. Using your logical schema from Part1-a, write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key. You can make reasonable assumptions regarding the attribute domains (note that uniformly setting every column to *VARCHAR2(100)* is not reasonable).

create table AUTHORS (

LastName varchar2(40) not null,

FirstName varchar2(40),

ID number(5),

BirthDate varchar2(20),

primary key (ID)

);

create table PUBLISHERS (

name varchar2(40),

PID number(5),

address varchar2(100),

primary key (PID)

);

create table BOOKS (

ISBN varchar2(30),

title varchar2(100),

publisherID number(5),

primary key (ISBN),

foreign key (publisherID) references PUBLISHERS(PID)

);

create table RANK (

AuthorID number(5),

ISBN varchar2(30),

rank\_no number(3),

primary key (ISBN, rank\_no),

foreign key (ISBN) references BOOKS(ISBN),

foreign key (AuthorID) references AUTHORS(ID)

);

1. Using logical schema from Part1-b write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key. For **Students** table, clearly state the assumptions you have made when choosing a primary key. You can make reasonable assumptions regarding the attribute domains.

create table ADVISORS (

ID number(10),

name varchar2(100),

address varchar2(300),

research\_area varchar2(100),

primary key (ID)

);

create table STUDENTS (

SID number(15),

firstname varchar2(50),

lastname varchar2(50) not null,

dob varchar2(20),

telephone varchar2(15),

AdvisorID number(10),

primary key (SID),

foreign key (AdvisorID) references ADVISORS(ID)

);

1. Write SQL INSERT statements to populate your database from Part1-a with the following data (NOTE: remember that strings would need to use single quotes, e.g., 'Asimov')
   1. (King, Stephen, 2, September 9 1947)
   2. (Asimov, Isaac, 4, January 2 1920)
   3. (Verne, Jules, 7, February 8 1828)
   4. (Rowling, Joanne, 37, July 31 1965)
   5. (Bloomsbury Publishing, 17, London Borough of Camden)
   6. (Arthur A. Levine Books, 18, New York City)
   7. (1111-111, Databases from outer space, 17)
   8. (2222-222, Dark SQL, 17)
   9. (3333-333, The night of the living databases, 18)
   10. (2, 1111-111, 1)
   11. (4, 1111-111, 2)
   12. (4, 2222-222, 2)
   13. (7, 2222-222, 1)
   14. (37, 3333-333, 1)
   15. (2, 3333-333, 2)

insert into AUTHORS

values ( 'King', 'Stephen', 2, 'September 9 1947' );

insert into AUTHORS

values ( 'Asimov', 'Isaac', 4, 'January 2 1920' );

insert into AUTHORS

values ( 'Verne', 'Jules', 7, 'February 8 1828' );

insert into AUTHORS

values ( 'Rowling', 'Joanne', 37, 'July 31 1965' );

insert into PUBLISHERS

values ( 'Bloomsbury Publishing', 17, 'London Borough of Camden' );

insert into PUBLISHERS

values ( 'Arthur A. Levine Books', 18, 'New York City' );

insert into BOOKS

values ( '1111-111', 'Databases from outer space', 17 );

insert into BOOKS

values ( '2222-222', 'Dark SQL', 17 );

insert into BOOKS

values ( '3333-333', 'The night of the living databases', 18 );

insert into RANK

values ( 2, '1111-111', 1 );

insert into RANK

values ( 4, '1111-111', 2 );

insert into RANK

values ( 4, '2222-222', 2 );

insert into RANK

values ( 7, '2222-222', 1 );

insert into RANK

values ( 37, '3333-333', 1 );

insert into RANK

values ( 2, '3333-333', 2 );

**Part 3**

You want to create a relation representing US presidents. Suppose that the following is true of the data you want to represent:

* No two presidents have the same name and year of birth
* No two presidents have the same inauguration date
* All presidents have a name, a year of birth, and have been inaugurated into office
* Not all presidents are affiliated with a political party

Write a valid create table statement for the relation.

create table Presidents (

firstname varchar2(100) not null,

lastname varchar2(100),

birthyear number(4),

inaug\_date date not null,

primary key (lastname, birthyear),

unique(inaug\_date)

);

Create table Political\_Party (

partyname varchar2(130),

lastname varchar2(100),

birthyear number(4),

primary key (lastname, birthyear),

foreign key (lastname, birthyear) references presidents(lastname, birthyear)

);

**Part 4**

Let R(ABCDEFGH) satisfy the following functional dependencies:

A → B, CH → A, B → E, BD → C, EG → H, DE → F.

Use transitive rule to find additional F.D.s that are satisfied by R?

**A 🡪 E**

**CH 🡪 B**

**CH 🡪 E**

**(DGA)+ 🡪 R (Candidate Key)**

**(DGB)+ 🡪 R (Candidate Key)**

**(DGAB)+ 🡪 R (Super Key)**

**(DGAC)+ 🡪 R (Super Key)**

**(DGAE)+ 🡪 R (Super Key)**

**(DGAH)+ 🡪 R (Super Key)**

**(DGBC)+ 🡪 R (Super Key)**

**(DGBE)+ 🡪 R (Super Key)**

**(DGBH)+ 🡪 R (Super Key)**

**(DGCE)+ 🡪 R (Candidate Key)**

**(DGCH****)+ 🡪 R (Candidate Key)**

**Part 5**

Consider a MEETING table that records information about meetings between clients and executives in the company. Each record contains the names of the client and the executive’s name as well as the office number, floor and the building. Finally, each record contains the city that the building is in and the date of the meeting. The table is in First Normal Form and the primary key is (Client, Office).

(Date, Client, Office, Floor, Building, City, Executive)

You are given the following functional dependencies:

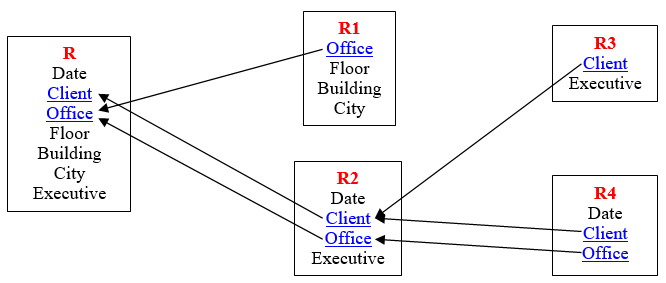
Building → City

Office → Floor, Building, City

Client → Executive

Client, Office → Date

1. Remove any existing partial dependencies and convert the logical schema to the Second Normal Form. Please remember that when performing schema decomposition, you need to denote primary key for every new table as well as the foreign key that will allow us to reconstruct the original data.



**R1**, **R3**, and **R4** are decomposed relations. They are in 2NF form because there is no partial dependency in these three tables.

1. Remove any existing transitive dependencies to create a set of logical schemas in Third Normal Form. Again, remember to denote primary keys and foreign keys (including which primary key those foreign keys point to).

The decomposed relations **R1**, **R3**, and **R4** in Part “a” are also in Third Normal Form because every attribute in these tables are determined only by the primary key of the table.

**Part 6**

Consider a table that stores information about students, student name, GPA, honors list and the credits that the student had completed so far.

(First, Last, GPA, Honor, Credits)

You are given the following functional dependencies

First, Last → GPA, Honor, Credits

GPA → Honor

1. Is this schema in Second Normal Form? If not, please state which FDs violate 2NF and decompose the schema accordingly.

**ANSWER**: Yes, this schema is already in 2NF because every attribute in the table is determined by the entire primary key (but not by any subset). There is no partial dependency.

1. Is this schema in Third Normal Form? If not, please state which FDs violate 3NF and decompose the schema accordingly.

**ANSWER:** No, this schema is not in 3NF because GPA 🡪 Honor. To get 3NF, we need to eliminate transitive dependencies. Every attribute in the table must be determined only by the primary key of the table.

