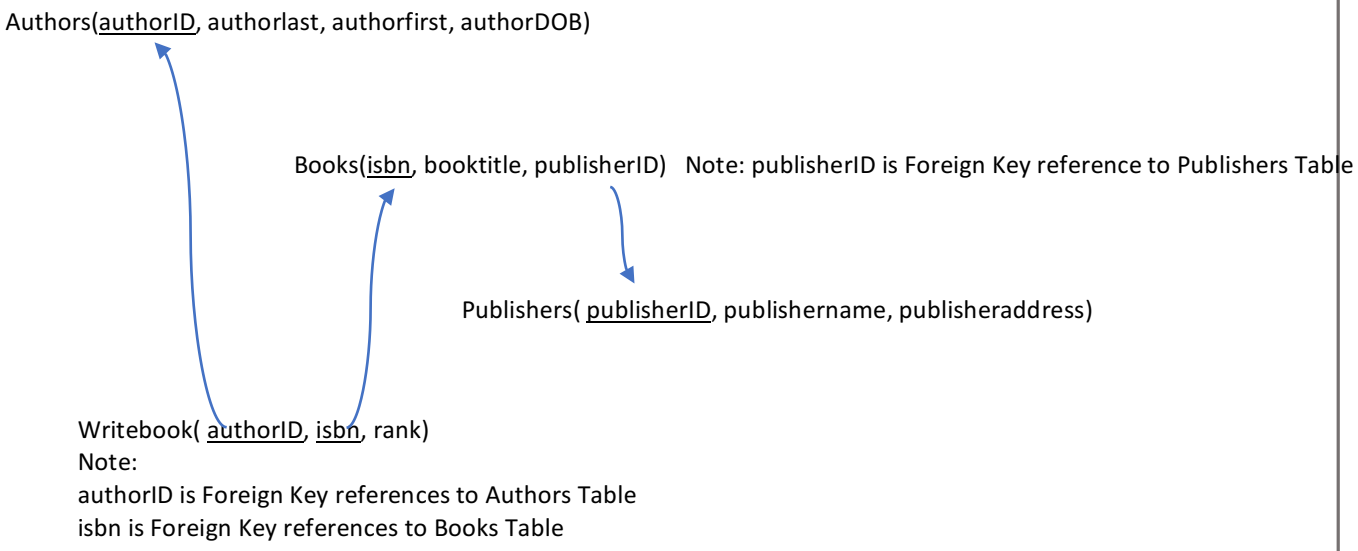
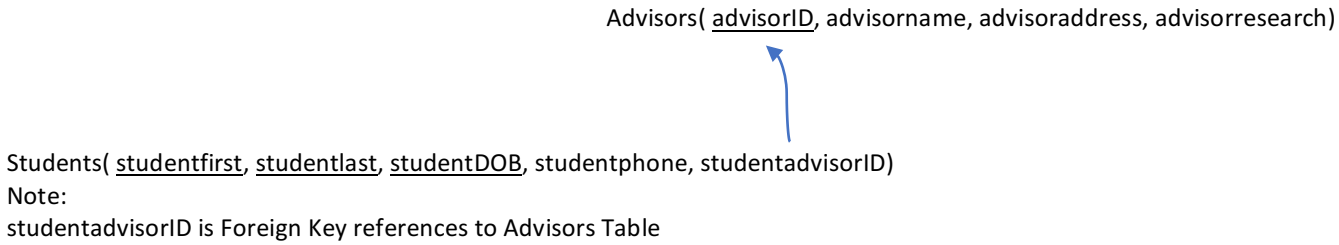
CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 1: Relational schema: Underline attribute is the Primary Key

a.) Schema Tables contain Authors, Books, Publishers and Writebook



b.) Schema Tables contain Students and Advisors (Underlined Attribute(s) is the Primary KEY)



Assume that students' First Name and Last Name might be the same among all the students, so that using Student's First Name (studentfirst), Last Name (studentlast) and DOB (studentDOB) as the Composite Primary Key to identify the student

CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 2 - 1: According part 1-a, write necessary SQL DDL to script to create the tables (Authors, Publishers, Books, Wirtebook)

Assumptions:

- The Authors table contains the authors' Last Name, Frist Name, Author's ID and Author's DOB
- The Authors' last name and first name might be the same among different authors, so that the Unique Primary key for this table is the authorID (identifies the Author)
- The Publishers table contains all publishers' name, publishers' ID and address
- The Publishers name might be the same among different publishers, so that publisherID (Unique Primary Key) is used to identify the publisher.
- The Books table contains every book's ISBN, Title and Publisher Ref ID
- The ISBN (Unique Primary Key) identifies the Books
- The PublisherID (Foreign Key) references to the Publisher identity from the Publishers Table.
- The Writebook table contains the Authors' ID, ISBN and ranking which represent the rank of author contributing to the book.
- Since Multiple Authors might contribute to ONE book, so that the authorID and ISBN (Composite Primary Key) identify the rank for that particular book.

```
/*CREATE Authors Table*/
CREATE TABLE Authors
(
 authorlast VARCHAR2(15)  NOT NULL,
 authorfirst VARCHAR2(15) NOT NULL,
 authorID NUMBER(2) NOT NULL,
 authorDOB DATE NOT NULL,

 CONSTRAINT Authers_PK
  PRIMARY KEY (authorID)
);


/*CREATE Publishers Table*/
CREATE TABLE Publishers
(
 publishername VARCHAR2(25) NOT NULL,
 publisherID NUMBER(2) NOT NULL,
 publisheraddress VARCHAR2(30),
```

```
  CONSTRAINT Publishers_PK
    PRIMARY KEY (publisherID)
);


/*CREATE Books Table*/
CREATE TABLE Books
(
  isbn VARCHAR2(8) NOT NULL,
  booktittle VARCHAR2(45) NOT NULL,
  publisherID NUMBER(2) NOT NULL,

  CONSTRAINT Books_PK
    PRIMARY KEY (isbn),

  CONSTRAINT Books_FK
    FOREIGN KEY (publisherID)
      REFERENCES Publishers (publisherID)
);


/*CREATE Wirtebook Table*/
CREATE TABLE Writebook
(
  authorID NUMBER(2) NOT NULL,
  isbn VARCHAR2(8) NOT NULL,
  rank NUMBER(2) NOT NULL,

  CONSTRAINT Writebook_PK
    PRIMARY KEY (authorID,isbn),

  CONSTRAINT Witebook_FK1
    FOREIGN KEY (isbn)
      REFERENCES Books (isbn),

  CONSTRAINT Witebook_FK2
    FOREIGN KEY (authorID)
      REFERENCES Authors (AuthorID)
);
```

CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 2 - 2: According part 1-b, write necessary SQL DDL to script to create the tables (Students, Advisors).

Assumption:

- The Students table contains all the students information including First Name, Last Name, DOB, Telephone and reference to their advisor
- Assume that students' First Name and Last Name might be the same among all the students, so that using Student's First Name (studentfirst), Last Name (studentlast) and DOB (studentDOB) as the Composite Primary Key to identify the student
- The student's Reference to their Advisor (studentadvisorID) is a Foreign Key to identify their advisor
- The Advisors table contains all the advisors' ID number, Name, Address and Research Area
- The ID (advisorID) of advisors is the Unique Primary Key identifies the Advisors.

```
/*CREATE Advisors Table*/
CREATE TABLE Advisors
(
 advisorID NUMBER(5) NOT NULL,
 advisorname VARCHAR2(50) NOT NULL,
 advisoraddress VARCHAR2(50) NOT NULL,
 advisorresearch VARCHAR2(50) NOT NULL,

 CONSTRAINT Advisors_PK
   PRIMARY KEY (advisorID)
);

/*CREATE Students Table*/
CREATE TABLE Students
(
 studentfirst VARCHAR2(25) NOT NULL,
 studentlast VARCHAR2(25) NOT NULL,
 studentDOB DATE NOT NULL,
 studentphone NUMBER(10) NOT NULL,
 studentadvisorID NUMBER(5) NOT NULL,

 CONSTRAINT Students_PK
  PRIMARY KEY(studentfirst, studentlast, studentDOB),

 CONSTRAINT Students_FK
```

Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

```
   FOREIGN KEY (studentadvisorID)
    REFERENCES Advisors(advisorID)
);
```

Part 2 - 3: According part 1-a, write SQL INSERT statements to populate the database

```
/* INSERT Values to Authors Table*/
```

a) INSERT INTO Authors VALUES('King','Stephen',2, TO_DATE('September 9 1947', 'MONTHDDYYYY'));
b) INSERT INTO Authors VALUES('Asimov','Isaac',4, TO_DATE('January 2 1921', 'MONTH dd yyyy'));
c) INSERT INTO Authors VALUES('Verne','Jules',7, TO_DATE('February 8 1828', 'MONTH dd yyyy'));
d) INSERT INTO Authors VALUES('Rowling','Joanne',37, TO_DATE('July 31 1965', 'MONTH dd yyyy'));

```
/* INSERT Values to Publishers Table*/
```

e) INSERT INTO Publishers VALUES('Bloomsbury Publishing',17 ,'London Borough of Camden');
f) INSERT INTO Publishers VALUES('Arthur A. Levine Books',18 ,'New York City');

```
/*INSERT Values to Books Table*/
```

g) INSERT INTO Books VALUES('1111-111', 'Databases from outer space', 17);
h) INSERT INTO Books VALUES('2222-222', 'Dark SQL', 17);
i) INSERT INTO Books VALUES('3333-333', 'The night of the living databases', 18);

```
/*INSERT Values to Writebook Table*/
```

j) INSERT INTO Writebook VALUES(2, '1111-111', 1);
k) INSERT INTO Writebook VALUES(4, '1111-111', 2);
l) INSERT INTO Writebook VALUES(4, '2222-222', 2);
m) INSERT INTO Writebook VALUES(7, '2222-222', 1);
n) INSERT INTO Writebook VALUES(37, '3333-333',1);
o) INSERT INTO Writebook VALUES(2, '3333-333', 2);

CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 3: Python function generates and return a SQL INSERT statement given a table name and value list as parameters:

Code:

```
[In]:
def generateInsert(tableName,inputList):
    newList=[]
    for item in inputList:
        if item.isdigit()== True:
            item=int(item)
            newList.append(item)
        else:
            newList.append(item)
    return "INSERT INTO {} VALUES ({}),".format(tableName, newList)

[In]:
generateInsert("Students", ["1","Jane","A-"])

[Out]:
    "INSERT INTO Students VALUES ([1, 'Jane', 'A-'])"

[In]:
generateInsert("Phones", ["42","312-555-1212"])

[Out]:
    "INSERT INTO Phones VALUES ([42, '312-555-1212'])"
```

CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 4
Building → City
Office → Floor, Building, City
Client → Executive
Client, Office → Date

a. Different buildings might exist in the same city. In this case, redundancy problem will be occurred. The following is the example of the redundancy might happen.

Building A	Chicago
Building B	Chicago
Building C	Chicago

b. 2nd Normal Form: Remove Partial Dependencies (Underlined is the Primary Key)
Office → Floor, Building, City
Client → Executive
Client, Office → Date
(Client is the Foreign Key references to the Client Table)
(Office is the Foreign Key references to the Office Table)

Relational Schema:

OFFICE (Office, Floor, Building, City) Note: Building is Foreign Key reference to Building Table

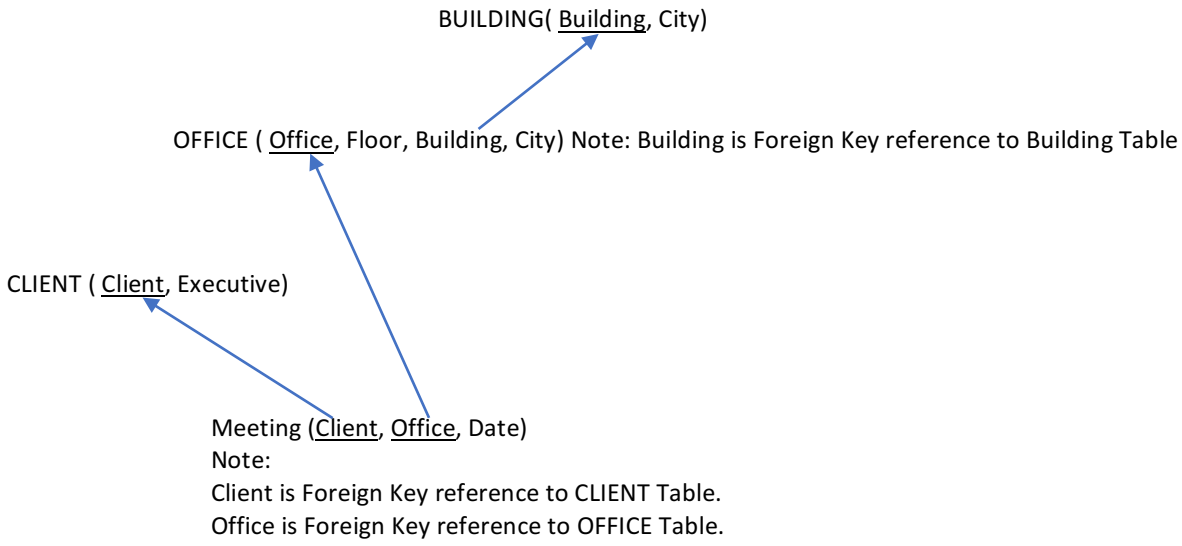
CLIENT (Client, Executive)

Meeting (Client, Office, Date)
Note:
Client is Foreign Key reference to CLIENT Table.
Office is Foreign Key reference to OFFICE Table.

Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

c. 3rd Normal Form: Remove Transitive Dependencies (Underlined is the Primary Key)
Building → City
Office → Floor, Building (Building is the Foreign Key ref. to the Building Table)
Client → Executive
Client, Office → Date
(Client is the Foreign Key references to the Client Table)
(Office is the Foreign Key references to the Office Table)

Relational Schema:



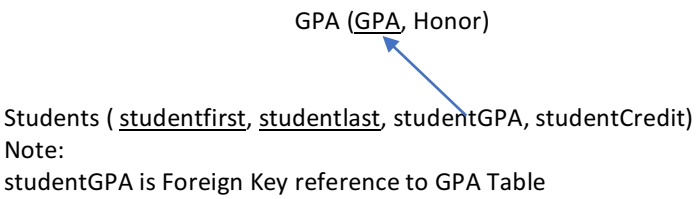
CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017

Part 5
First, Last → GPA, Honor, Credits
GPA → Honor

a Yes. The above (First, Last → GPA, Honor, Credits) is Second Normal Form because it does NOT contain partial functional dependencies.

b NO. It is NOT Third Normal Form because FD (First, Last → GPA, Honor, Credits) which contains transitive functional dependencies (GPA, Honor) violates the 3NF rule. The schema should be decomposed as the following:

First, Last → GPA, Credits
GPA → Honor



CSC 455 HW #1
Kevin Kai Chung, Ying
Email: kying@mail.depaul.edu
Date: 04-10-2017