Alex Teboul
Assignment 1

CSC 455: Database Processing for Large-Scale Analytics
Due ~~Wednesday, April 17th~~ → Sunday, April 14th
*****Sorry about the late submission mixed up the due dates with another course.**

Part 1

Write a python function that is going to generate and return a SQL INSERT statement given a table name and value list as parameters. For example,
print(generateInsert('Students', ['1', 'Jane', 'A+'])) should print
INSERT INTO Students VALUES (1, Jane, A+);

```
#Alex Teboul
#DSC 450: A1
#Part 1
def generateInsert(table, values):
    ''' Accepts a table name as a string parameter and values as a list parameter, returns
    this in the form "INSERT INTO {table} VALUES {list};'''
    formatted_values = ', '.join(values)
    result = 'INSERT INTO {} VALUES ({});'.format(table, formatted_values)
    return result
print(generateInsert('Students', ['1', 'Jane', 'A+']))
print(generateInsert('Phones', ['42', '312-555-1212']))
```

OUTPUT:

```
(base)alexs-mbp:~alexteboul$/anaconda3/bin/python/Users/alexteboul/Desktop/A1_DSC450_AlexTeboul.py
INSERT INTO Students VALUES (1, Jane, A+);
INSERT INTO Phones VALUES (42, 312-555-1212);
```

- *works

Part 2

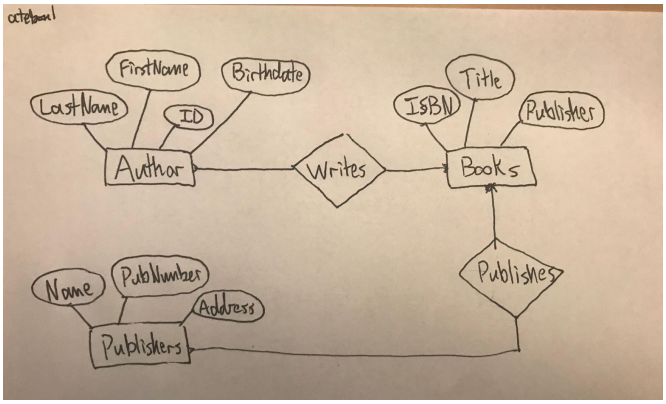
a) Define a relational schema with underlined (primary) keys and arrows connecting foreign keys and primary keys for a database containing the following information. We will do a similar example in the beginning of the next lecture.

· **Authors** have LastName, FirstName, ID, and Birthdate (identified by ID)

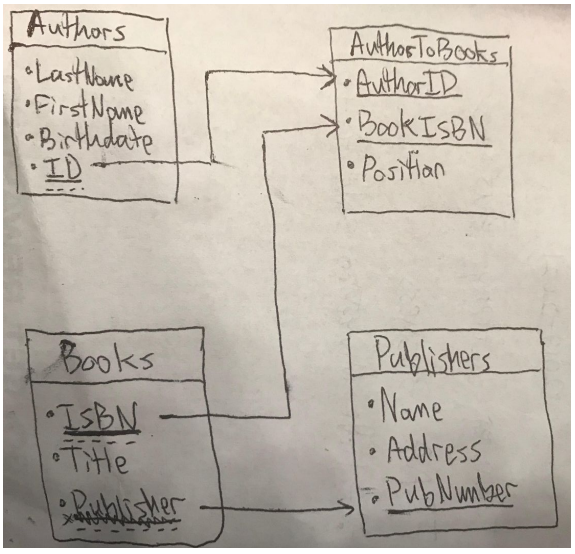
· **Publishers** have Name, PubNumber, Address (identified by PubNumber)

· **Books** have ISBN, Title, Publisher (each book has a publisher and is identified by its ISBN).

· Authors **Write** Books; since many authors can co-author a book, we need to know the relative contribution of the author to a book, signified by their position in the author list (i.e. 1, 2, 3, etc.).



- We should have a separate table like AuthorToBooks containing the AuthorID, ISBN of the associated book, and the Position of that author in the author list. The new table is defined by the AuthorID and ISBN.



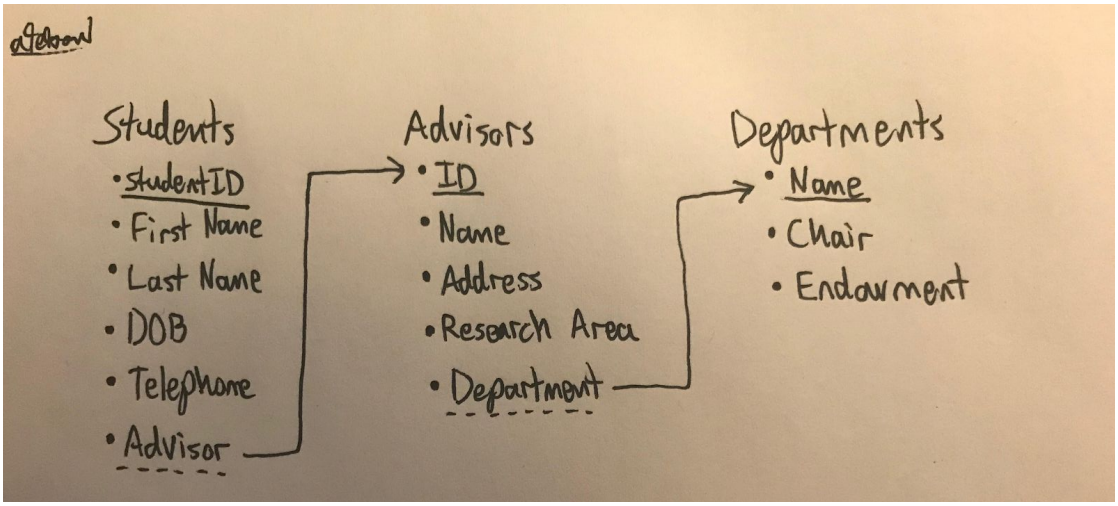
- **Primary Keys:** Authors(ID), Publishers(PubNumber), Books(ISBN), AuthorsToBooks(AuthorID,ISBN)
- **Foreign Keys**: Books(Publisher), Authors(ID)

b) Define a relational schema for students, student advisors, and advisor departments

· **Students** have StudentID, First Name, Last Name, DOB, Telephone and a reference to their advisor

· **Advisors** have ID, Name, Address, Research Area, and a reference link to their Department

· **Departments** have Name, Chair, Endowment (identified by Name)



- **Primary Keys:** Students(StudentID), Advisors(ID), Departments(Name)
- **Foreign Keys**: Students(Advisor) which links Advisors(ID), and Advisors(Department) which links to Departments(Name).
- The reference to a student's advisor should be to the advisor's ID so it's unique, the department can just be the department name because there shouldn't be overlap there - question even states it's unique.

Part 3

1) Using your logical schema from Part 2-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 (e.g., setting every column to *VARCHAR2(100)* is not reasonable).

```
CREATE TABLE Authors (
   LastName VARCHAR2(16),
   FIRSTNAME VARCHAR2(16),
   ID NUMBER(7),
   Birthdate VARCHAR (20),
```

```
  CONSTRAINT Authors_PK
    PRIMARY KEY (ID),

  CONSTRAINT Authors_FK
    FOREIGN KEY (ID)
      REFERENCES AuthorsToBooks(AuthorID)
);

CREATE TABLE Books (
  ISBN VARCHAR2(15),
  Title VARCHAR2(100),
  Publisher NUMBER(7),

  CONSTRAINT Books_PK
    PRIMARY KEY (ISBN, Publisher),

  CONSTRAINT Books_FK1
    FOREIGN KEY (ISBN)
      REFERENCES AuthorsToBooks(BookISBN),

  CONSTRAINT Books_FK2
    FOREIGN KEY (Publisher)
      REFERENCES Publishers(PubNumber)
);

CREATE TABLE Publishers (
  Name VARCHAR2(50),
  PubNumber NUMBER(7),
  Address VARCHAR2(50),

  CONSTRAINT Publishers_PK
    PRIMARY KEY (PubNumber)
);

CREATE TABLE AuthorsToBooks(
  AuthorID NUMBER(7),
  BooksISBN ISBN VARCHAR2(15),
  Position NUMBER(5),
```

```
  CONSTRAINT AuthorsToBooks_PK
    PRIMARY KEY (AuthorID, BooksISBN)
);
```

2) Using logical schema from Part 2-b write the necessary SQL DDL script to create the tables. Be sure to specify every primary key and every foreign key.

```
CREATE TABLE Students (
  StudentID VARCHAR2(9),
  FirstName VARCHAR2(16),
  LastName VARCHAR2(16),
  DOB DATE,
  Telephone VARCHAR2(13),
  Advisor VARCHAR2(9),

  CONSTRAINT Students_PK
    PRIMARY KEY (StudentID),

  CONSTRAINT Students_FK
    FOREIGN KEY (Advisor)
      REFERENCES Advisors(ID)
);

CREATE TABLE Advisors(
  ID VARCHAR2(9),
  Name VARCHAR2(32),
  Address VARCHAR2(40),
  ResearchArea VARCHAR2(50),
  Department VARCHAR2(32),

  CONSTRAINT Advisors_PK
    PRIMARY KEY (ID),

  CONSTRAINT Advisors_FK
    FOREIGN KEY (Department)
      REFERENCES Departments(Name)
);
```

```
CREATE TABLE Departments(
   Name VARCHAR2(32),
   Chair VARCHAR2(32),
   Endowment NUMBER(12)

   CONSTRAINT Department_PK
      PRIMARY KEY (Name)

);
```

3) Write SQL INSERT statements to populate your database from Part 2-a with the following data (NOTE: remember that strings would need to use single quotes, e.g., 'Asimov'). Be sure to verify that your statements worked correctly and loaded the data.

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

--Publishers
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')

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

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

Part 4

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)

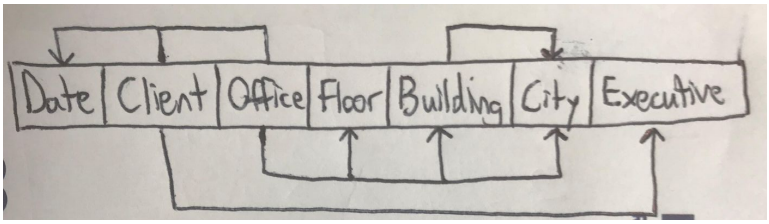
You are given the following functional dependencies:
Building → City
Office → Floor, Building, City
Client → Executive
Client, Office → Date

1. For the functional dependency Building → City, explain the redundancy problem and possible consequences through an example (you can make up your own building names as you see fit).
    a. The redundancy problem is that every time a meeting takes place in a particular building, the table brings up the Building and the City, when that building is unique to the city. This is redundant, simply having the building alone should be sufficient and a separate Building and City table can be linked to the main meetings table.
    b. Ex.

Date	Client	Office	Floor	Building	City	Executive
4/14/19	Alex	101	1	CDM	Chicago	Obama
4/14/19	Bill	101	1	CDM	Chicago	Obama
4/14/19	Frank	405	4	Starbucks	Chicago	Oprah

    c. For the Building → City functional dependency, CDM is always in Chicago, so we don't need the City column in this table, just the Building is sufficient.

2. 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.
    a. Functional dependencies:



b. Three tables:

Offices(<u>Office</u>, Floor, Building, City)

Primary = Office ; Foreign = Office

Clients(<u>Client</u>, Executive)

Primary = Client ; Foreign = Client

ClientOffice(<u>ClientRef</u>, <u>OfficeRef</u>, Date)

Primary = ClientRef and OfficeRef ; Foreign = none

c. The Building → City can be dealt with for 3NF beause it's a transitive dependency and with the above 3 tables the partial dependencies are already eliminated.

3. 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).
   a. Now we deal with the Building → City transitive dependency by making a new table.
   b. Four tables:

Buildings(<u>Building</u>, City)

Primary = Building ; Foreign = none

Offices(<u>Office</u>, Floor, Building)

Primary = Office ; Foreign = Building and Office [linked to ClientOffice(OfficeRef)]

Clients(<u>Client</u>, Executive)

Primary = Client ; Foreign = Client [linked to ClientOffice(ClientRef)]

ClientOffice(<u>ClientRef</u>, <u>OfficeRef</u>, Date)

Primary = ClientRef and OfficeRef ; Foreign = none

Part 5

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.
    a. **Yes, we're already in 2NF** just like the Name Address Zip code example from class. Specifically, no partial dependencies exist.
    b. That said, there is redundancy in that certain GPAs will always give Honor list status. In the same way that certain Zip codes always reference particular cities. To deal with this redundancy we need to decompose further.

1. Is this schema in Third Normal Form? If not, please state which FDs violate 3NF and decompose the schema accordingly.
    a. **No, not yet in 3NF** because of the redundancy in GPA/Honors (transitive dependancy between GPA&Honor). The 3NF decomposition of the schema is shown below. It's just making sure there is a 4 column First, Last, GPA, Credits table (for which First,Last are primary keys and GPA is a foreign key) and a GPA, Honor table (for which GPA is the primary key).
    b.
    First, Last → GPA, Credits
    GPA → Honor

Be sure that your name and "Assignment 1" appear at the top of your submitted file.