Christian Craig
DSC 450

DSC 450: Database Processing for Large-Scale Analytics
Assignment Module 1

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. We have not covered SQL yet, so you don't need to run the insert statements, just get your python code to return the required string:

Chose the below example:

For example,
print(generateInsert('Students', ['1', 'Jane', 'A+'])) should print
INSERT INTO Students VALUES (1, Jane, A+);

All statements start from **INSERT INTO**, followed by the name of the table, followed by **VALUES** and the comma-separated list of supplied values inside parenthesis, terminated by a semi-colon
Make sure that your function returns the string rather than prints it.

```
>>> def generateInsert(table,lst_vals):
...     '''
...     function that generates and returns a  sql insert
...     statement,given a table and a list as parameters
...     '''
...     sql_insert = "INSERT INTO {} VALUES ({});".format(table,str(lst_vals).replace('[','').replace(']',''))
...     if type(table)!=str or type(lst_vals)!=list: #table must be string and lst_vals be a list
...         print("Input of table or lst_vals is incorrect. Please re-enter values and remember:\n1. table needs to be a string\n2. ls
...     else:
...         return sql_insert
...
>>> print(generateInsert('Students', ['1', 'Jane', 'A+']))
INSERT INTO Students VALUES ('1', 'Jane', 'A+');
>>> print(generateInsert('Phones',['42','312-555-1212']))
INSERT INTO Phones VALUES ('42', '312-555-1212');
```

Christian Craig
DSC 450

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.

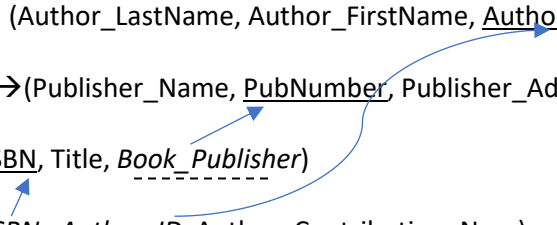
- **Authors** have LastName, FirstName, ID, and Birthdate (identified by ID)
- **Publishers** have Name, PubNumber, Address (identified by PubNumber)
- **Books** have ISBN, Title, Publisher (each book has a publisher and is identified by its ISBN).
- Authors **Write** Books; since many authors can co-author 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.)

Authors_Table → (Author_LastName, Author_FirstName, Author_ID, and Author_Birthdate)

Publishers_Table→(Publisher_Name, PubNumber, Publisher_Address)

Books_Table→(ISBN, Title, *Book_Publisher*)

Write_Table→(*ISBN* , *Author_ID*, Author_Contribution_Num)



*Per class discussion group → Book_Publisher = PubNumber

The Primary key for the Authors_Table is Author_ID. The primary key for the Publishers_Table is PubNumber. The Primary Key for Books_Table is ISBN. The foreign key comes from the Books_Table, specifically *Book_Publisher.* Two foreign keys come from the Write_Table, which are ISBN (connecting to the Books_Table) and Author_ID (connecting to the Authors_Table).

Christian Craig
DSC 450

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)

Students_Table→(Student_ID, FirstName, LastName, DOB, Phone, *Advisors_Ref*)

Advisors_Table→(Advisors_ID, Adv_Name, Address, Research_Area, *Dept_Ref*)

Departments_Table→(Dept_Name, Chair, Endowment)

In the Students_Table, Student_ID is the primary key. In the Advisors_Table, Advisors_ID is the primary key. In the Departments_Table, Dept_Name is the primary key. *Advisors_Ref* is the foreign key connecting Students_Table and Advisors_Table. *Dept_Ref* is foreign key connecting the Advisors_Table to the Departments_Table.