COMS W4111-002/V02, Spring 22: Take Home Final

Overview

Instructions

- There is an [Ed discussion thread (https://edstem.org/us/courses/18760/discussion/1464598)](https://edstem.org/us/courses/18760/discussion/1464598) that will specify completion and submission instructions.

Due Date, Completing the Exam and Rules

1. The final exam is due at 11:59 PM on Monday, 08-May-2022. **You are not allowed to use late days.**

2. You may use on-line information and sources to answer questions. But,
   A. You cannot simply cut and paste answers or code. Your answer must demonstrate that you understood the material and are capable of producing an answer from your understanding.
   B. You must cite any sources of information that you used. This can simply be a comment in a text/markdown cell in your answer. For example, `(Note: I used https://www.w3schools.com/sql/sql_check.asp to help me with the syntax for adding a check constraint).`
   C. You do NOT need to cite lecture notes, recordings, slides, ... You do not need to cite information from the recommended textbook or textbook slides.

3. You **MUST NOT** collaborate with ANYONE, including other students. You MAY speak with the professor or a TA to discuss the exam.

4. If you have questions, post them as PRIVATE question on Ed discussion and use the Category `Exams->Final`.

5. There is a pinned Ed discussion thread [Midterm Clarifications (https://edstem.org/us/courses/18760/discussion/1464595)](https://edstem.org/us/courses/18760/discussion/1464595) that the professor and TA will use to communicate updates and clarifications. **Students are responsible for checking this post.**

6. Individual questions provide explanations for how to complete the question.

Environment Setup and Test

1. This section tests your environment.

2. You will need to change the URLs and password in some of the cells below to match your configuration.

3. You may need to load data and copy databases. The relevant questions provide information.

4. **You must read the comments in the setup cells. You will need objects and functions in the cells for some questions.**

MySQL

In [29]:
```
%load_ext sql
```

```
The sql extension is already loaded. To reload it, use:
  %reload_ext sql
```

In [30]:
```
# Connect to MySQL. Replace dbuser:dbuserdbuser with your MySQL user ID and passw
%sql mysql+pymysql://root:Edy990127@localhost
```

Out[30]:
```
'Connected: root@None'
```

In [31]:
```
from sqlalchemy import create_engine
```

In [32]:
```
# Create an SQL Alchemy engine for using MySQL. Replace dbuser:dbuserdbuser with
sql_engine = create_engine("mysql+pymysql://root:Edy990127@localhost")
```

In [33]:
```
import pandas as pd
```

In [34]:
```
import pymysql
```

```
In [35]: def get_mysql_connection(user_id, password, autocommit=True):
             """
             Creates and return a connection to the local MySQL database.

             :param user_id: The user ID for the connection.
             :param password: Corresponding password.
             :param autocommit: Should this connection use autocommit for executed stateme
                 for explanation:
                 - https://pymysql.readthedocs.io/en/latest/modules/connections.html
                 - https://dev.mysql.com/doc/refman/5.6/en/innodb-autocommit-commit-rollba
             """
             sql_conn = pymysql.connect(
                 user= 'root',
                 password= 'Edy990127',
                 host="localhost",
                 port=3306,
                 cursorclass=pymysql.cursors.DictCursor,
                 autocommit=autocommit)

             return sql_conn


         def close_connection(sql_connection):
             """
             Closes a connection.

             :param sql_connection: The connection to close.
             """
             if sql_connection:
                 sql_connection.close()
```

- The following cell(s) load information about CU courses.

```
In [36]: # You must set these parameters to the values for you final submission and databa
         # Replace dff9 with your UNI
         #
         mysql_url = "mysql+pymysql://root:Edy990127@localhost"
         final_schema = "de2418_s22_final"
         course_info_table = "course_info"
         course_info_file = "./course_info.json"

         def read_and_save_json_file(file_name, schema_name, table_name):

             df = pd.read_json(file_name)
             eng = create_engine(mysql_url)
             df.to_sql(table_name, schema=schema_name, con=eng, index=False, if_exists="re
```

```
In [37]: %sql create database if not exists de2418_s22_final
```

```
 * mysql+pymysql://root:***@localhost
1 rows affected.
```

```
Out[37]: []
```

In [38]:
```
read_and_save_json_file(course_info_file, final_schema, course_info_table)
```

- The following cell tests your load. You must replace the schema name with your schema.

In [39]:
```
%sql use $final_schema
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
```

Out[39]:
```
[]
```

```
In [40]: %%sql
             select
                 callnumber, schoolcode, coursetitle, coursesubtitle,
                     instructor1name, typename,
                     course, prefixname,
                     enrollmentstatus,
                     maxsize, numenrolled,
                     callnumber,
                     prefixlongname,
                     departmentcode, campuscode,
                     divisionname, departmentname,
                     typecode, term,
                     schoolname, meets1
         from course_info
         where instructor1name like "%Ferguson%Donald%"
         order by callnumber;
```

```
 * mysql+pymysql://root:***@localhost
7 rows affected.
```

Out[40]:

callnumber	schoolcode	coursetitle	coursesubtitle	instructor1name	typename	cou
11009	SEAS	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	FERGUSON, DONALD F	LECTURE	COMS4111W
11038	SEAS	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	FERGUSON, DONALD F	LECTURE	COMS6156E
12423	SEAS	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	FERGUSON, DONALD F	LECTURE	COMS4111W
13390	SEAS	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	FERGUSON, DONALD F	LECTURE	COMS6156E
13911	SEAS	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	FERGUSON, DONALD F	LECTURE	COMS4111W
16112	SEAS	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	FERGUSON, DONALD F	LECTURE	COMS6156E
18408	SEAS	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	FERGUSON, DONALD F	LECTURE	COMS4111W

MongoDB

```
In [41]: mongodb_url = "mongodb+srv://de2418:Edy990127@aaaa.1raes.mongodb.net/GoT?retryWri
In [42]: from pymongo import MongoClient
```

```
In [43]: mongo_client = MongoClient(mongodb_url)
```

```
In [44]: # Note that your list of database will be different.
         # You should see the databases from your MongoDB Atlas account.
         #
         dbs = mongo_client.list_databases()
         list(dbs)
```

```
Out[44]: [{'name': 'GoT', 'sizeOnDisk': 552960, 'empty': False},
          {'name': 'admin', 'sizeOnDisk': 344064, 'empty': False},
          {'name': 'local', 'sizeOnDisk': 7151849472, 'empty': False}]
```

Neo4j

```
In [45]: #
         # You must replace the following with your AuraDB Neo4j information.
         neo_url = "neo4j+s://13f30c6c.databases.neo4j.io:7687"
         neo_auth = ("neo4j", "isZCQgcAMZ9UusAH_w-hoqAtB7TsjJnXiEwrvbuCbGA")
```

```
In [46]: from py2neo import data, Graph, NodeMatcher, Node, Relationship, RelationshipMatc
```

```
In [47]: neo_graph = Graph(neo_url, auth=neo_auth)
```

```
In [48]: #
         # The following code assumes that you have loaded the sample movie database in Ne
         tom = NodeMatcher(neo_graph).match(name="Tom Hanks")
         for t in tom:
             print(dict(t))
```

Written Questions

W1: Serializability

- Briefly define *conflict serializable.* Relative to *conflict* serializable, *define _conflicting operations.*

Answer:

Conflict-serializability of a schedule is the property that it can be transformed from a non-serial into a serial schedule by swapping non-conflicting operations.

Conflicting operations are operations that satisfy the following conditions:

1. Operations belong to different transactions
2. Operations operate on the same data item
3. At Least one of operations is a write operation

- Is the following schedule *conflict serializable?* Why?

T1	**T2**
R(A)	
R(B)	
	R(A)
	R(B)
	W(B)
W(A)	

Answer:

The above schedule is not conflict serializable, because when I swap the non-conflicting operations. I cannot get a serial schedule.

W2: Locking

- Briefly define *Two-Phase Locking* and *Strict Two-Phase Locking.* What condition does *Strict Two-Phase Locking* prevent?

Answer:

Two-Phase Locking is a process that divides the locking and unlocking into two phases, the growing phase and shrinking phase. By doing so, two-phase locking ensures conflict serializable schedules.

Strict Two-Phase Locking is an extension to basic two-phase locking that a transaction must hold all its exclusive locks till it commits/aborts. By doing so, Strict Two-Phase Locking snsures recoverability and avoids cascading roll-backs.

Strict Two-Phase Locking prevent cascading roll-back.

Type *Markdown* and LaTeX: α^2

W3: Metadata

- Define *metadata.* Give an example of metadata for a SQL database. What is the name of the database/schema where MySQL stores its metadata?

Answer:

Metadata is data that provides information about other data.

Metadata for a SQL database could be the information about the database, database objects, database files.

Information schema is the name of the schema where MySQL stores its metadata

W4: Locking

- Briefly define *Two-Phase Locking* and *Strict Two-Phase Locking.* What condition does *Strict Two-Phase Locking* prevent?

Answer:

Two-Phase Locking is a process that divides the locking and unlocking into two phases, the growing phase and shrinking phase. By doing so, two-phase locking ensures conflict serializable schedules.

Strict Two-Phase Locking is an extension to basic two-phase locking that a transaction must hold all its exclusive locks till it commits/aborts. By doing so, Strict Two-Phase Locking snsures recoverability and avoids cascading roll-backs.

Strict Two-Phase Locking prevent cascading roll-back.

W5: Functional Dependency

- Briefly define *functional dependency.* What changes to a schema do you make to eliminate functional dependencies.

Answer:

Functional Dependency is a constraint that determines the relation of one attribute to another attribute in a Database Management System.

We decompose a schema into BCNF(Boyce-Codd Normal Form) to eliminate functional dependencies.

Type *Markdown* and LaTeX: α^2

W6: Entity Types

- Briefly define *associative entity* and *weak entity.*

Answer:

Associative entity is an entity that is placed between two entities to break up a M:N relationship. Since the associative entity does not have an existence of its own, it is actually a weak entity as well, but will always be called an associative entity.

Weak Entity is an entity that exists to store multi-valued attributes of a strong entity. That is, it doesn't sit between two independent entities.

Type *Markdown* and LaTeX: α^2

W7: RAID

- Briefly define and explain the differences between *RAID 0, RAID 1* and *RAID 5.*

Answer:

For RAID 0 and RAID 1, minumum number of drive is 2. For RAID 5, minimum number of drive is 3. Read and Write performance is RAID 0 better than RAID 1 better than RAID 5. RAID 0 is generally used for high end workstations, data logging, real-time rendering. RAID 1 is generally used for operating systems, transaction databases. RAID 5 is generally used for data warehousing, web serving, archiving.

Type *Markdown* and LaTeX: α^2

W8: Column versus Row Storage

- Briefly explain *row-oriented storage* versus *column-oriented storage.* Explain a scenario/database query pattern where column-oriented storage is beneficial.

Answer:

Column oriented database stores and retrieves data in columns and hence it can only read the relevant data if required. Row oriented database stores and retrieves data one row at a time and hence could read unnecessary data if some of the data in a row are required. Records in row-oriented storage are easier to read and write than column-oriented storage.

SELECT SUM(balance_amount) from table is better for column-oriented storage.

Type *Markdown* and LaTeX: α^2

W9: Indexes

- Briefly define:
  - Sparese index
  - Dense index
  - Clustered index

Answer:

For dense index, each record in the main table strictly has one entry in the index table. For sparse index, index records are not created for every search key, which means that the index is not based on keys. For clustered index, data file is ordered on a non-key field

Type *Markdown* and LaTeX: α^2

W10: Query Processing

- Briefly define:
  - Access path
  - Join order optimization
  - Equivalent queries/expressions.

Answer:

Access path is the location Microsoft DOS or Windows looks when a command is not an internal command or in the current directory.

Join order optimization is the case that find the more preferable order of join that could make the join require least amoun of work.

The goal of query optimization is to reduce the system resources required to fulfill a query, and ultimately provide the user with the correct result set faster. If equivalent queries are provided, then query optimization process will be much easier to execute.

Data Modeling Question

- Consider the [IMDB datasets. (https://www.imdb.com/interfaces/)](https://www.imdb.com/interfaces/)

- The following queries run on my laptop show the structure of some of the data. These queries will not run on your computer. So, if you execute the cell, you will lose the out.

In [13]:
```
%sql select * from imdb_raw_fast.name_basics limit 10;
```

```
 * mysql+pymysql://dbuser:***@localhost
10 rows affected.
```

Out[13]:

nconst	primaryName	birth_year	death_year	primaryProfession	
nm0000001	Fred Astaire	1899	1987	soundtrack,actor,miscellaneous	tt0053137,tt005
nm0000002	Lauren Bacall	1924	2014	actress,soundtrack	tt0038355,tt011
nm0000003	Brigitte Bardot	1934	None	actress,soundtrack,music_department	tt0049189,tt005
nm0000004	John Belushi	1949	1982	actor,soundtrack,writer	tt0072562,tt007
nm0000005	Ingmar Bergman	1918	2007	writer,director,actor	tt0060827,tt005
nm0000006	Ingrid Bergman	1915	1982	actress,soundtrack,producer	tt0034583,tt003
nm0000007	Humphrey Bogart	1899	1957	actor,soundtrack,producer	tt0033870,tt003
nm0000008	Marlon Brando	1924	2004	actor,soundtrack,director	tt0070849,tt006
nm0000009	Richard Burton	1925	1984	actor,soundtrack,producer	tt0057877,tt008
nm0000010	James Cagney	1899	1986	actor,soundtrack,director	tt0031867,tt004

In [14]:
```
%sql select * from imdb_raw_fast.title_basics limit 10;
```

```
 * mysql+pymysql://dbuser:***@localhost
10 rows affected.
```

Out[14]:

tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	
tt0000001	short	Carmencita	Carmencita	0	1894	\N	1	
tt0000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	\N	5	
tt0000003	short	Pauvre Pierrot	Pauvre Pierrot	0	1892	\N	4	Animatio
tt0000004	short	Un bon bock	Un bon bock	0	1892	\N	12	
tt0000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	\N	1	
tt0000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	\N	1	
tt0000007	short	Corbett and Courtney Before the Kinetograph	Corbett and Courtney Before the Kinetograph	0	1894	\N	1	
tt0000008	short	Edison Kinetoscopic Record of a Sneeze	Edison Kinetoscopic Record of a Sneeze	0	1894	\N	1	
tt0000009	short	Miss Jerry	Miss Jerry	0	1894	\N	40	
tt0000010	short	Leaving the Factory	La sortie de l'usine Lumière à Lyon	0	1895	\N	1	

In [15]:
```
%sql select * from imdb_raw_fast.title_principals limit 10;
```

```
 * mysql+pymysql://dbuser:***@localhost
10 rows affected.
```

Out[15]:

tconst	ordering	nconst	category	job	characters
tt0000001	1	nm1588970	self	\N	["Self"]
tt0000001	2	nm0005690	director	\N	\N
tt0000001	3	nm0374658	cinematographer	director of photography	\N
tt0000002	1	nm0721526	director	\N	\N
tt0000002	2	nm1335271	composer	\N	\N
tt0000003	1	nm0721526	director	\N	\N
tt0000003	2	nm1770680	producer	producer	\N
tt0000003	3	nm1335271	composer	\N	\N
tt0000003	4	nm5442200	editor	\N	\N
tt0000004	1	nm0721526	director	\N	\N

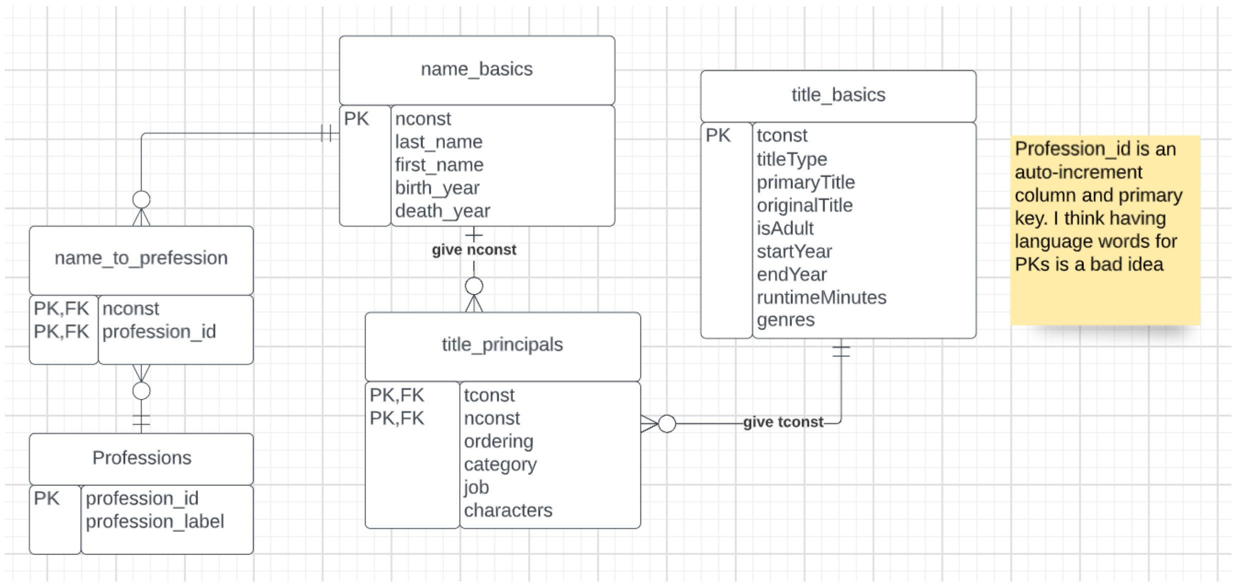
- Given this information and the descriptions of the data (https://www.imdb.com/interfaces/), produce a Crow's Foot diagram of a data model you would use to hold the data.
- Your design will require transforming the data, which you do not need to do. But, you should do a good data model. That is, define a "to be" data model.

Answer:

In [1]:
```
er_model_file_name = 'lucidchart.png'

print("\n")
from IPython.display import Image
Image(filename=er_model_file_name)
```

Out[1]:



Type *Markdown* and LaTeX: α^2

Data Loading and Transformation

- In the setup for MySQL, you tested the setup by loading information from the Columbia Open Data Service. (https://opendataservice.columbia.edu/)

In [50]:
```
%%sql
    select
        *
from course_info
where instructor1name like "%Ferguson%Donald%"
order by callnumber;
```

```
 * mysql+pymysql://root:***@localhost
7 rows affected.
```

Out[50]:

Instructor3Name	DivisionCode	ChargeAmt1	SchoolCode	CourseSubtitle	Meets5	Approval	Instr
	IF		SEAS	INTRODUCTION TO DATABASES			F
	EP		SEAS	CLOUD COMPUTING			F
	IF		SEAS	INTRODUCTION TO DATABASES			F
	EP		SEAS	CLOUD COMPUTING			F
	IF		SEAS	INTRODUCTION TO DATABASES			F
	EP	395.00	SEAS	CLOUD COMPUTING			F
	IF	395.00	SEAS	INTRODUCTION TO DATABASES			F

- The columns in the table are:

```
In [26]: %sql show columns from de2418_s22_final.course_info;
```

```
 * mysql+pymysql://root:***@localhost
44 rows affected.
```

Out[26]:

Field	Type	Null	Key	Default	Extra
Instructor3Name	text	YES		None	
DivisionCode	text	YES		None	
ChargeAmt1	text	YES		None	
SchoolCode	text	YES		None	
CourseSubtitle	text	YES		None	
Meets5	text	YES		None	
Approval	text	YES		None	
Instructor1Name	text	YES		None	
Meets6	text	YES		None	
NumFixedUnits	text	YES		None	
TypeName	text	YES		None	
SubtermCode	text	YES		None	
ClassNotes	text	YES		None	
Meets3	text	YES		None	
Course	text	YES		None	
PrefixName	text	YES		None	
EnrollmentStatus	text	YES		None	
MaxSize	text	YES		None	
NumEnrolled	text	YES		None	
ChargeMsg1	text	YES		None	
CallNumber	bigint	YES		None	
BulletinFlags	text	YES		None	
SubtermName	text	YES		None	
PrefixLongname	text	YES		None	
Meets4	text	YES		None	
DepartmentCode	text	YES		None	
CampusCode	text	YES		None	
DivisionName	text	YES		None	
CourseTitle	text	YES		None	
DepartmentName	text	YES		None	
ExamDate	text	YES		None	
ChargeMsg2	text	YES		None	
CampusName	text	YES		None	

Field	Type	Null	Key	Default	Extra
ChargeAmt2	text	YES		None	
Instructor2Name	text	YES		None	
MaxUnits	bigint	YES		None	
TypeCode	text	YES		None	
Term	bigint	YES		None	
Instructor4Name	text	YES		None	
SchoolName	text	YES		None	
MinUnits	bigint	YES		None	
Meets1	text	YES		None	
Meets2	text	YES		None	
ExamMeet	text	YES		None	

- For the purposes of the final, you only need to consider the columns in the following query.

```
In [56]: %%sql
    select
        Instructor3Name,
        DivisionCode,
        SchoolCode,
        CourseSubtitle,
        Approval,
        Instructor1Name,
        TypeName,
        ClassNotes,
        Course,
        PrefixName,
        EnrollmentStatus,
        MaxSize,
        NumEnrolled,
        CallNumber,
        BulletinFlags,
        PrefixLongname,
        DepartmentCode,
        CampusCode,
        DivisionName,
        CourseTitle,
        DepartmentName,
        CampusName,
        Instructor2Name,
        TypeCode,
        Term,
        Instructor4Name,
        SchoolName
    from
        course_info
        limit 10;
```

```
 * mysql+pymysql://root:***@localhost
10 rows affected.
```

Out[56]:

Instructor3Name	DivisionCode	SchoolCode	CourseSubtitle	Approval	Instructor1Name	TypeNa
	BU	SBUS	Financial Accounting		SOURSOURIAN, JESSICA	LECTU
	BU	SBUS	Accounting I: Financial A		ZIV, AMIR	LECTU
	BU	SBUS	Financial Accounting		ZIV, AMIR	LECTU
	BU	SBUS	Exemption: Accounting I - Fina		FACULTY, .	LECTU
	BU	SBUS	Financial Accounting		BREUER, MATTHIAS	LECTU

Instructor3Name	DivisionCode	SchoolCode	CourseSubtitle	Approval	Instructor1Name	TypeNa
	BU	SBUS	Financial Accounting		BREUER, MATTHIAS	LECTU
	BU	SBUS	Financial Accounting		ZIV, AMIR	LECTU
	BU	SBUS	Financial Accounting		ZIV, AMIR	LECTU
	BU	SBUS	Financial Accounting		CAI, WEI	LECTU
	BU	SBUS	Financial Accounting		CAI, WEI	LECTU

- Your task is to "clean up" the data. This will involve:
    - Splitting/decomposing into multiple tables.
    - Setting types for columns.
    - Creating constraints and indexes, etc.
- In the following cells, include and execute:
    - The create and alter table statements.
    - The query statements for copying and transforming the data.

Answer:

```
In [ ]: #
        # All of your DDL statements must go here. You must group all statements for a ta
        # with one cell containing statements for a table.
        #
```

In [75]:
```
%%sql
drop table if exists de2418_s22_final.course_dimension;
create table de2418_s22_final.course_dimension
(
    Course           varchar(20) default '0' not null,
    CourseTitle      text                    null,
    CourseSubtitle   text                    null,
    CallNumber       text                    null,
    Approval         varchar(50)             null,
    TypeName         varchar(50)             null,
    TypeCode         varchar(50)             null,
    ClassNotes       text                    null,
    PrefixName       varchar(50)             null,
    EnrollmentStatus varchar(50)             null,
    MaxSize          varchar(50)             null,
    NumEnrolled      varchar(50)             null,
    BulletinFlags    text                    null,
    PrefixLongname   text                    null,
    term             varchar(20) default '0' not null,
    Instructor1Name  text                    null,
    Instructor2Name  text                    null,
    Instructor3Name  text                    null,
    Instructor4Name  text                    null,
    primary key (Course, term)
);

create index course_dimension_Course_index
    on de2418_s22_final.course_dimension (Course);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
0 rows affected.
```

Out[75]:
```
[]
```

In [76]:
```
%%sql
drop table if exists de2418_s22_final.department_dimension;
create table de2418_s22_final.department_dimension
(
    SchoolCode     varchar(50) default '0' not null,
    SchoolName     text                    null,
    DepartmentCode varchar(20) default '0' not null,
    DepartmentName text                    null,
    CampusCode     varchar(20) default '0' not null,
    CampusName     text                    null,
    DivisionCode   varchar(20) default '0' not null,
    DivisionName   text                    null,
    primary key (SchoolCode, DivisionCode, DepartmentCode, CampusCode)
);

create index department_dimension_index
    on de2418_s22_final.department_dimension (SchoolCode, CampusCode, DepartmentC
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[76]: []

In [82]:
```
%%sql
drop table if exists de2418_s22_final.course_fact;
create table de2418_s22_final.course_fact
(
    Course         varchar(20) default '0' not null,
    SchoolCode     varchar(20) default '0' not null,
    CampusCode     varchar(20) default '0' not null,
    DepartmentCode varchar(20) default '0' not null,
    DivisionCode   varchar(20) default '0' not null,
    term           varchar(20) default '0' not null,
    primary key (Course, SchoolCode, CampusCode, DepartmentCode, DivisionCode, te
    constraint course_fact_course_dimension_Course_term_fk
        foreign key (Course, term) references course_dimension (Course, term),
    constraint course_fact_fk
        foreign key (SchoolCode, CampusCode, DepartmentCode, DivisionCode) refere
);
```

```
 * mysql+pymysql://root:***@localhost
0 rows affected.
0 rows affected.
```

Out[82]: []

In [78]:
```
#
#
# Put your cleanup and load statements in the following cells.
#
```

In [79]:
```
%%sql
insert into de2418_s22_final.department_dimension (SchoolCode,
                                                    SchoolName, DepartmentCode, De
                                                    CampusCode, CampusName, Divisi
select distinct SchoolCode, SchoolName, DepartmentCode, DepartmentName, CampusCod
from de2418_s22_final.course_info
```

```
 * mysql+pymysql://root:***@localhost
406 rows affected.
```

Out[79]:
```
[]
```

In [80]:
```
%%sql
insert into de2418_s22_final.course_dimension (Course,CourseTitle,CourseSubtitle,
                                               ClassNotes,PrefixName,EnrollmentSt
                                               PrefixLongname,term,Instructor1Nam
                                               Instructor3Name,Instructor4Name)
select Course,CourseTitle,CourseSubtitle,CallNumber,Approval,TypeName,TypeCode,
                                               ClassNotes,PrefixName,EnrollmentSt
                                               PrefixLongname,term,Instructor1Nam
                                               Instructor3Name,Instructor4Name
from de2418_s22_final.course_info
```

```
 * mysql+pymysql://root:***@localhost
24140 rows affected.
```

Out[80]:
```
[]
```

In [83]:
```
%%sql
insert into de2418_s22_final.course_fact (Course, SchoolCode, CampusCode, Departm
select Course, SchoolCode, CampusCode, DepartmentCode, DivisionCode, term from de
```

```
 * mysql+pymysql://root:***@localhost
24140 rows affected.
```

Out[83]:
```
[]
```

In [86]:
```
# Write some SQL queries that show information about Donald Ferguson and the cour
#
```

```
In [87]: %%sql
         select * from de2418_s22_final.course_dimension
         where instructor1name like "%Ferguson%Donald%"
         order by callnumber;
```

```
 * mysql+pymysql://root:***@localhost
7 rows affected.
```

Out[87]:

Course	CourseTitle	CourseSubtitle	CallNumber	Approval	TypeName	TypeCode	Cl
COMS4111W003	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	11009		LECTURE	LC	
COMS6156E001	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	11038		LECTURE	LC	
COMS4111W002	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	12423		LECTURE	LC	
COMS6156E001	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	13390		LECTURE	LC	
COMS4111W002	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	13911		LECTURE	LC	
COMS6156EV01	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	16112		LECTURE	LC	N S
COMS4111WV02	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	18408		LECTURE	LC	N S

In [94]:
```
%%sql
select * from
(select * from de2418_s22_final.course_dimension
where instructor1name like "%Ferguson%Donald%") as a
join de2418_s22_final.course_fact as b
on a.Course = b.Course and a.term = b.term
```

```
 * mysql+pymysql://root:***@localhost
7 rows affected.
```

Out[94]:

Course	CourseTitle	CourseSubtitle	CallNumber	Approval	TypeName	TypeCode	Cl
COMS4111W002	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	12423		LECTURE	LC	
COMS4111W002	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	13911		LECTURE	LC	
COMS4111W003	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	11009		LECTURE	LC	
COMS4111WV02	INTRODUCTION TO DATABASES	INTRODUCTION TO DATABASES	18408		LECTURE	LC	N S
COMS6156E001	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	13390		LECTURE	LC	
COMS6156E001	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	11038		LECTURE	LC	
COMS6156EV01	TOPICS IN SOFTWARE ENGINEERING	CLOUD COMPUTING	16112		LECTURE	LC	N S

NoSQL

Introduction

- We studied inheritance/generalization/specialization when data modeling.
- We modeled `person, student, employee` in a HW. The folder containing the notebook has CSV files with data for employee and student.

In [95]:
```
student_df = pd.read_csv('./students.csv')
employee_df = pd.read_csv('./employee.csv')
```

In [96]:
```
student_df.head(10)
```

Out[96]:

	Unnamed: 0	id	first_name	middle_name	last_name	email	u
0	0	103edf18-a882-11ec-a3ad-edb7cbd2bb34	Zared	NaN	Fenelon	afenelona@themeforest.net	Z
1	1	103f6b22-a882-11ec-a3ad-edb7cbd2bb34	Bibbye	Annabal	Guesford	aguesfordb@tumblr.com	BA(
2	2	10402f4e-a882-11ec-a3ad-edb7cbd2bb34	Duffy	Auberon	Pounder	apounder2h@reuters.com	DAI
3	3	10406b1c-a882-11ec-a3ad-edb7cbd2bb34	Base	NaN	Baybutt	bbaybutty@tmall.com	BI
4	4	1040de26-a882-11ec-a3ad-edb7cbd2bb34	Jenine	Berry	Habberjam	bhabberjam2k@examiner.com	JBI
5	5	104153d8-a882-11ec-a3ad-edb7cbd2bb34	Reece	Corbett	Caps	ccapsz@telegraph.co.uk	RC(
6	6	10419168-a882-11ec-a3ad-edb7cbd2bb34	Barry	Cullin	Elias	celias1k@scribd.com	BCI
7	7	104203d2-a882-11ec-a3ad-edb7cbd2bb34	Kerwin	Conrade	Foort	cfoortw@vinaora.com	KC
8	8	1042383e-a882-11ec-a3ad-edb7cbd2bb34	Brigida	Cameron	Maclean	cmaclean13@mac.com	BCI
9	9	10426bc4-a882-11ec-a3ad-edb7cbd2bb34	Woodrow	Camile	Moughtin	cmoughtin17@illinois.edu	WCI

In [97]:
```
employee_df.head(10)
```

Out[97]:

	Unnamed: 0	id	first_name	middle_name	last_name	email
0	0	103e6a74-a882-11ec-a3ad-edb7cbd2bb34	Sanders	Arline	Breckell	abreckell1x@fotki.com
1	1	103f2a54-a882-11ec-a3ad-edb7cbd2bb34	Ethelin	NaN	Fidele	afidele12@google.ru
2	2	103fa696-a882-11ec-a3ad-edb7cbd2bb34	Xenia	Ardella	Kief	akieft@free.fr
3	3	103fe1ec-a882-11ec-a3ad-edb7cbd2bb34	Cari	Andriana	Leask	aleask1n@devhub.com
4	4	10409f88-a882-11ec-a3ad-edb7cbd2bb34	Lemmy	Burr	Bradnocke	bbradnockek@nifty.com
5	5	10411828-a882-11ec-a3ad-edb7cbd2bb34	Sibylle	Bearnard	Lalley	blalley2d@rediff.com
6	6	1041cc50-a882-11ec-a3ad-edb7cbd2bb34	Lu	Cinnamon	Flaxman	cflaxman1b@cdbaby.com
7	7	1043122c-a882-11ec-a3ad-edb7cbd2bb34	Hobart	Dominic	Croal	dcroalx@purevolume.com
8	8	10434364-a882-11ec-a3ad-edb7cbd2bb34	Marylin	Darcy	Favey	dfavey2p@mozilla.com
9	9	1043a598-a882-11ec-a3ad-edb7cbd2bb34	Ailbert	Danie	Warmisham	dwarmishame@soundcloud.com

- For the purposes of the final, you only need to worry about:
  - uni
  - first_name, middle_name, last_name
  - email
  - employee_type

- enrollment_year

Neo4j

- The task is to load the data into Neo4j. You do not need to worry about relationships. This question is solely about creating nodes.
- The question is focusing on how you decide to represent inheritance, e.g. that the nodes are either
  - Person and Student
  - Person and Employee
- Put your code for loading the data in the cells below.

In [123]:
```
employee_df.employee_type = 'Employee'
```

In [125]:
```
person_df = student_df.append(employee_df)
```

In [126]:
```
person_df.reset_index(inplace = True, drop = True)
```

In [131]:
```
person_df.employee_type.fillna('Student', inplace = True)
```

In [140]:
```
person_df['full_name'] = person_df['first_name'] + person_df['last_name']
```

In [165]:
```
student_df['full_name'] = student_df['first_name'] + student_df['last_name']
employee_df['full_name'] = employee_df['first_name'] + employee_df['last_name']
```

In [166]:
```
person_df.drop_duplicates(subset = ['full_name'], inplace = True)
person_df.reset_index(inplace = True, drop = True)
student_df.drop_duplicates(subset = ['full_name'], inplace = True)
student_df.reset_index(inplace = True, drop = True)
employee_df.drop_duplicates(subset = ['full_name'], inplace = True)
employee_df.reset_index(inplace = True, drop = True)
```

In [168]:
```
from py2neo import data, Graph, NodeMatcher, Node, Relationship, RelationshipMatc
g = Graph('neo4j+s://13f30c6c.databases.neo4j.io:7687',
         auth = ("neo4j", "isZCQgcAMZ9UusAH_w-hoqAtB7TsjJnXiEwrvbuCbGA"))
```

In [169]:
```
cypher_q1 = "create (c:GoT_personName {personName: $name}) return c"
```

In [170]:
```
for i in range(len(person_df)):
    result = g.run(cypher_q1, name = person_df.full_name[i])
```

In [171]:
```
cypher_q4 = "create (c:GoT_studentName {studentName: $name}) return c"
for i in range(len(student_df)):
    result = g.run(cypher_q4, name = student_df.full_name[i])
```

In [179]:
```
cypher_q5 = "create (c:GoT_employeeName {employeeName: $name}) return c"
for i in range(len(employee_df)):
    result = g.run(cypher_q5, name = employee_df.full_name[i])
```

In [173]:
```
cypher_q2 = """
        match (s:GoT_studentName {studentName: $s_name}),
                (t:GoT_personName {personName: $p_name})
        create (s)-[:"""

cypher_q3 = """]->(t)"""
```

In [174]:
```
for i in range(len(student_df)):
    g.run(cypher_q2 + person_df.employee_type[i] + cypher_q3,
                   s_name = student_df.full_name[i],
                   p_name = person_df.full_name[i])
```

In [180]:
```
cypher_q2 = """
        match (s:GoT_employeeName {employeeName: $e_name}),
                (t:GoT_personName {personName: $p_name})
        create (s)-[:"""

cypher_q3 = """]->(t)"""
```

In [181]:
```
len(student_df)
```

Out[181]:
```
56
```

In [182]:
```
for i in range(len(employee_df)):
    g.run(cypher_q2 + person_df.employee_type[i+56] + cypher_q3,
                   e_name = employee_df.full_name[i],
                   p_name = person_df.full_name[i+56])
```

- Put code that shows a few nodes in the cells below.

In [206]:
```
cypher_qq = """
match (s:GoT_employeeName {employeeName: 'SandersBreckell'})-[r]->(t) return s,r,
"""

result = g.run(cypher_qq)
result = list(result)
```

In [207]:
```
simple_r = []
for r in result:
    simple_r.append(
        {
            "person": r['s']['employeeName'],
            "employment_type": ",".join(list(set(r['r'].types())))
        }
    )
```

In [208]:
```
simple_r_df = pd.DataFrame(simple_r)
simple_r_df
```

Out[208]:

	person	employment_type
0	SandersBreckell	Employee

In [210]:
```
cypher_qq = """
match (s:GoT_studentName {studentName: 'DougyBurchett'})-[r]->(t) return s,r,t
"""

result = g.run(cypher_qq)
result = list(result)
```

In [211]:
```
simple_r = []
for r in result:
    simple_r.append(
        {
            "person": r['s']['studentName'],
            "employment_type": ",".join(list(set(r['r'].types())))
        }
    )
```

In [212]:
```
simple_r_df = pd.DataFrame(simple_r)
simple_r_df
```

Out[212]:

	person	employment_type
0	DougyBurchett	Student

MongoDB

- The task is the same for MongoDB.
- You need to figure out how to model inheritance in a document DB. Specifically, how to model the fact that an entity is either both a Person-Employee or Person-Student.

- Put your code to load the data here.

```
In [ ]:
```

- Put some simple queries that show you structure below.

```
In [ ]:
```