ipython-sql Magic

```
In [1]: import json
```

```
In [2]: import pandas
```

```
In [3]: %load_ext sql
```

```
In [4]: %sql mysql+pymysql://root:dbuserbdbuser@localhost
```

```
In [5]: %sql select * from db_book.student limit 10
```

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

Out[5]:

ID	name	dept_name	tot_cred
00128	Zhang	Comp. Sci.	102
12345	Shankar	Comp. Sci.	32
19991	Brandt	History	80
23121	Chavez	Finance	110
44553	Peltier	Physics	56
45678	Levy	Physics	46
54321	Williams	Comp. Sci.	54
55739	Sanchez	Music	38
70557	Snow	Physics	0
76543	Brown	Comp. Sci.	58

PyMySQL

```
In [6]: import pymysql
In [7]: conn = pymysql.connect(
            host="localhost",
            user="root",
            password="dbuserbdbuser",
            cursorclass=pymysql.cursors.DictCursor,
            autocommit=True)
```

```
In [8]: cur = conn.cursor()
        res = cur.execute("select * from db_book.student limit 10")
        res = cur.fetchall()
```

```
In [9]: res
```

```
Out[9]: [{'ID': '00128',
          'name': 'Zhang',
          'dept_name': 'Comp. Sci.',
          'tot_cred': Decimal('102')},
         {'ID': '12345',
          'name': 'Shankar',
          'dept_name': 'Comp. Sci.',
          'tot_cred': Decimal('32')},
         {'ID': '19991',
          'name': 'Brandt',
          'dept_name': 'History',
          'tot_cred': Decimal('80')},
         {'ID': '23121',
          'name': 'Chavez',
          'dept_name': 'Finance',
          'tot_cred': Decimal('110')},
         {'ID': '44553',
          'name': 'Peltier',
          'dept_name': 'Physics',
          'tot_cred': Decimal('56')},
         {'ID': '45678',
          'name': 'Levy',
          'dept_name': 'Physics',
          'tot_cred': Decimal('46')},
         {'ID': '54321',
          'name': 'Williams',
          'dept_name': 'Comp. Sci.',
          'tot_cred': Decimal('54')},
         {'ID': '55739',
          'name': 'Sanchez',
          'dept_name': 'Music',
          'tot_cred': Decimal('38')},
         {'ID': '70557',
          'name': 'Snow',
          'dept_name': 'Physics',
          'tot_cred': Decimal('0')},
         {'ID': '76543',
          'name': 'Brown',
          'dept_name': 'Comp. Sci.',
          'tot_cred': Decimal('58')}]
```

Pandas and SQLAlchemy

In [16]:
```
import numpy as np
```

In [17]:
```
import sqlalchemy
```

In [18]:
```
engine = sqlalchemy.create_engine("mysql+pymysql://root:dbuserbdbuser@localhost")
```

In [19]:
```
df = pandas.read_sql("select * from db_book.student limit 10", con=engine)
```

In [20]:
```
df
```

Out[20]:

	ID	name	dept_name	tot_cred
0	00128	Zhang	Comp. Sci.	102.0
1	12345	Shankar	Comp. Sci.	32.0
2	19991	Brandt	History	80.0
3	23121	Chavez	Finance	110.0
4	44553	Peltier	Physics	56.0
5	45678	Levy	Physics	46.0
6	54321	Williams	Comp. Sci.	54.0
7	55739	Sanchez	Music	38.0
8	70557	Snow	Physics	0.0
9	76543	Brown	Comp. Sci.	58.0

MongoDB

Note: The following cell only works for me. I use this approach to avoid putting passwords in publicly accessible documents,

In [153...
```
import sys
import pymongo


# sys.path.append(
#     "/Users/donaldferguson/Dropbox/00Spring2023/Intro_to_Databases_S23/DONOTSHARE"
# )
```

In [154…

```
# import mongo_secrets

# mongo_url = mongo_secrets.mongo_atlas_url
password = "bq2150"
url = f"mongodb+srv://bq2150:{password}@s23-w4111.ovdrkzr.mongodb.net/?retryWrites=true&w=majority"
```

In [155…

```
mongo_client = pymongo.MongoClient(url)
# db = client.test
```

In [156…

```
list(mongo_client.list_databases())
```

Out[156]:

```
[{'name': 'S23_GoT', 'sizeOnDisk': 405504, 'empty': False},
 {'name': 's23_hw4', 'sizeOnDisk': 835584, 'empty': False},
 {'name': 'sample_airbnb', 'sizeOnDisk': 55152640, 'empty': False},
 {'name': 'sample_analytics', 'sizeOnDisk': 9674752, 'empty': False},
 {'name': 'sample_geospatial', 'sizeOnDisk': 1425408, 'empty': False},
 {'name': 'sample_guides', 'sizeOnDisk': 40960, 'empty': False},
 {'name': 'sample_mflix', 'sizeOnDisk': 49238016, 'empty': False},
 {'name': 'sample_restaurants', 'sizeOnDisk': 6946816, 'empty': False},
 {'name': 'sample_supplies', 'sizeOnDisk': 1196032, 'empty': False},
 {'name': 'sample_training', 'sizeOnDisk': 52195328, 'empty': False},
 {'name': 'sample_weatherdata', 'sizeOnDisk': 2932736, 'empty': False},
 {'name': 'admin', 'sizeOnDisk': 344064, 'empty': False},
 {'name': 'local', 'sizeOnDisk': 22173671424, 'empty': False}]
```

Neo4j

Question 8: Neo4j

I scoped my query to released dates after 2008. I will accept answers that are not scoped. I am not looking for perfection and am focusing on understanding the concepts.

Note: The following cells only work for me.

In [145…

```
# import neo4j_secrets
```

In [21]:

```
# aura_url = neo4j_secrets.aura_url
# aura_user = neo4j_secrets.aura_user
# aura_pw = neo4j_secrets.aura_pw
```

In [1]:
```
import py2neo
```

- Uncomment and set the Aura information, then run the test.

In [18]:
```
from py2neo import Graph

# aura_url =
# aura_user =
# aura_pw =

aura_url = 'neo4j+s://ea59b107.databases.neo4j.io'
aura_user = 'neo4j'
aura_pw = 'IspP3KUjmy_QSIZHnQg2eh5VRvDdSpeHjGGEEK7R8CY'

def t1():
    graph = Graph(aura_url, auth=(aura_user, aura_pw))
    q = "match (r:Person) where r.name='Tom Hanks' return r"
    res = graph.run(q)

    for r in res:
        print(r)
```

In [19]:
```
t1()
```

```
Node('Person', born=1956, name='Tom Hanks')
```

Relational Algebra

- Just kidding.
- I think we all have had as much fun as we can stand using relational algebra and the RelaX calculator.
- You're welcome.

Entity Relationship Modeling

Definition to Model

- The model you will diagram has four entity types:
    1. Faculty has the properties:
        - UNI
        - last_name
        - first_name
        - job_title
    2. Department has two properties:
        - department_code
        - department_name
    3. Student has the properties:
        - UNI
        - last_name
        - first_name
        - enrollment_year
    4. Section:
        - section_id
        - semester
        - year
        - credits

- The model has the following relationships:
    1. Faculty_Department:
        - A faculty may be associated with one or more departments.
        - The association has a type: member, chair, emeritus.
    2. Student_Department:
        - A student has exactly one department that is the major_department.

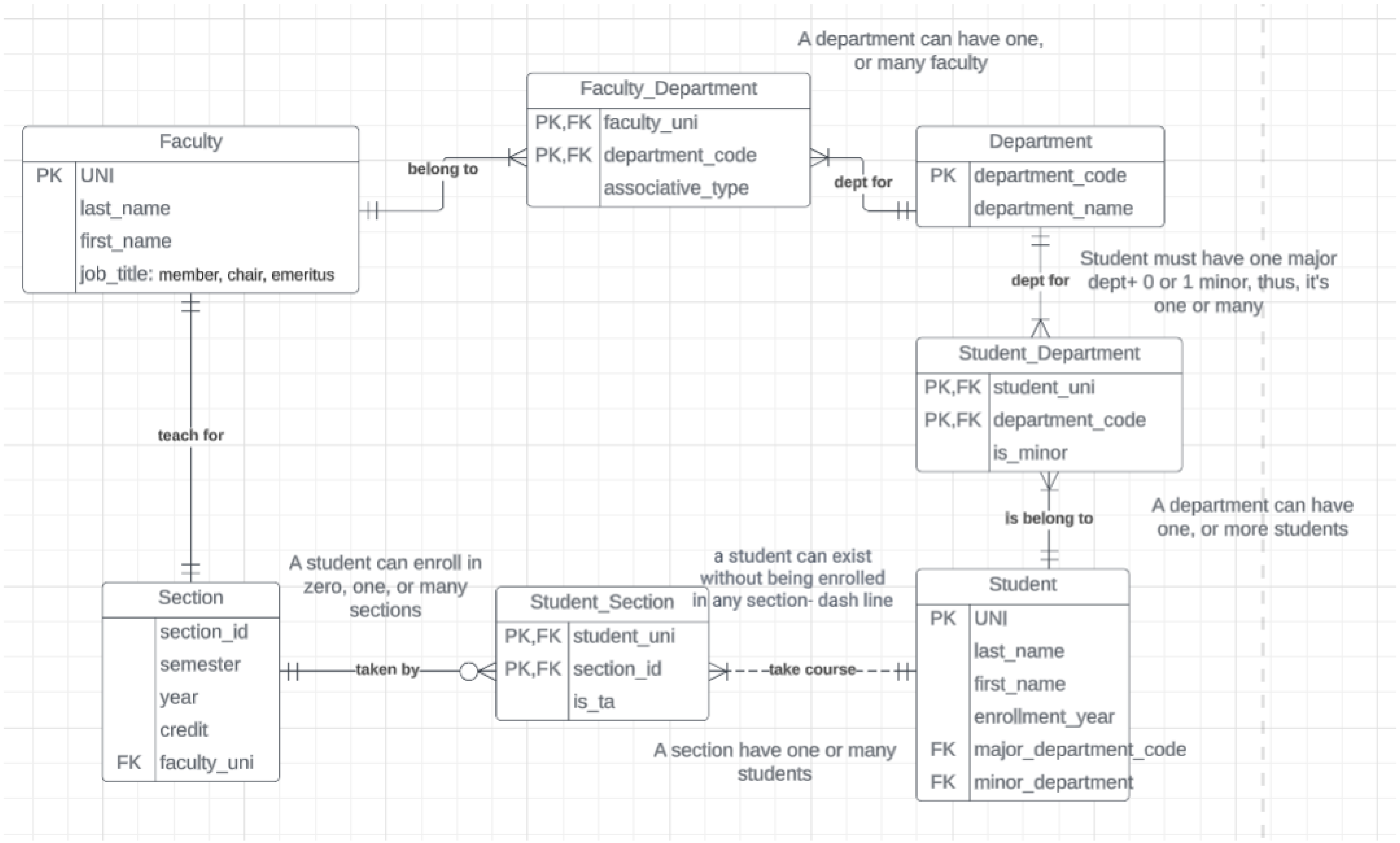
    - The student may have 0 or one minor_department.
3. Student_Section:
    - A student has a relationship to 0, 1 or many sections.
    - The student may be enrolled_in_ the section or a ta_for the section.
    - A section may have many enrolled students and many TAs.
4. Faculty_Section:
    - A faculty member teaches exactly one section per semester.
    - A section has exactly one instructor per semester.

- Use Lucidchart to draw a Crow's Foot Notation ER diagram for the logical model. You may add notes to explain any reasonable assumptions you make.

ER Diagram

In [20]:
```python
from IPython.display import Image
Image(filename='atlas-2.png')
```

Out[20]:



Model to Schema

- Create a new schema `s23_final_exam`.
- Implement and execute the DDL statements to implement your ER diagram.
- The university is extremely large. So, you should define indexes that you think appropriate.

SQL DDL Statements

In [40]:
```
#create schema
%sql create schema s23_final_exam
```

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

Out[40]:
```
[]
```

In [41]:
```
%%sql
use s23_final_exam;

drop table if exists Faculty;
create table Faculty (
  UNI VARCHAR(10) PRIMARY KEY,
  last_name VARCHAR(50) not null,
  first_name VARCHAR(50) not null,
  job_title VARCHAR(50)
);

drop table if exists Department;
create table Department (
  department_code VARCHAR(10) PRIMARY KEY,
  department_name VARCHAR(50) not null
);

drop table if exists Student;
create table  Student (
  UNI VARCHAR(10) PRIMARY KEY,
  last_name VARCHAR(50) not null,
  first_name VARCHAR(50) not null,
  enrollment_year INT not null,
  major_department_code VARCHAR(10) not null,
  minor_department_code VARCHAR(10),
  FOREIGN KEY (major_department_code) REFERENCES Department (department_code),
  FOREIGN KEY (minor_department_code) REFERENCES Department (department_code)
);

drop table if exists Section;
create table Section (
  section_id INT PRIMARY KEY,
  semester VARCHAR(20) not null,
  year INT not null,
  credits INT not null,
  faculty_UNI VARCHAR(10) not null,
  FOREIGN KEY (faculty_UNI) REFERENCES Faculty (UNI)
);
```

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

Out[41]:
```
[]
```

In [42]:
```
%%sql
use s23_final_exam;
drop table if exists Student_Section;
create table Student_Section (
  student_UNI VARCHAR(10) not null,
  section_id INT not null,
  is_ta BOOL not null,
  PRIMARY KEY (student_UNI, section_id),
  FOREIGN KEY (student_UNI) REFERENCES Student (UNI),
  FOREIGN KEY (section_id) REFERENCES Section (section_id)
);

drop table if exists Faculty_Department;
create table Faculty_Department (
  faculty_UNI VARCHAR(10) not null,
  department_code VARCHAR(10) not null,
  association_type enum('member', 'chair', 'emeritus') not null,
  PRIMARY KEY (faculty_UNI, department_code),
  FOREIGN KEY (faculty_UNI) REFERENCES Faculty (UNI),
  FOREIGN KEY (department_code) REFERENCES Department (department_code)
);

drop table if exists Student_Department;
create table Student_Department (
  student_UNI VARCHAR(10) not null,
  department_code VARCHAR(10) not null,
  is_minor BOOL not null,
  PRIMARY KEY (student_UNI, department_code),
  FOREIGN KEY (student_UNI) REFERENCES Student (UNI),
  FOREIGN KEY (department_code) REFERENCES Department (department_code)
);
```

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

Out[42]:
```
[]
```

In [43]:
```
%%sql
use s23_final_exam;
create index idx_section_instructor_UNI ON Section (faculty_UNI);
Ccreate index idx_student_section_section_id ON Student_Section (section_id);
create index idx_faculty_department_department_code ON Faculty_Department (department_code);
create index idx_student_department_department_code ON Student_Department (department_code);
```

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

Out[43]:
```
[]
```

SQL

Customer Summary

- The following is a view that is a summary of customers and orders from Classic Models.

In [107…
```
# #load in csv, just for take a look in datagrip

customer_order = pandas.read_csv("./customer_order_summary.csv")
customer_order.to_sql("customer_a", schema="s23_final_exam", con=engine,
                      index=False, if_exists="replace")
```

Out[107]:
```
326
```

In [108…
```
# create table customer_order_summary
%%sql
```