COMS W 4111-002
W4111 - Introduction to Databases
Section 003/V03, Fall 2022
Take Home Final

Exam Instructions

- We will publish instructions on Ed.

Environment Setup and Test

MySQL

- Replace `root` and `dbuserdbuser` for the correct values for you MySQL instance from previous homework assignments and exams.
- You will need the sample database (https://www.db-book.com/university-lab-dir/sample_tables-dir/index.html) that comes with the recommended textbook to execute the setup test.
  - You should have already installed the database because you need for previous assignments.
  - I named my database

In [1]:

```
%load_ext sql
```

In [2]:

```
%sql mysql+pymysql://root:dbuserdbuser@localhost
```

In [3]:

```
%sql select * from db_book.student
```

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

Out[3]:

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
76653	Aoi	Elec. Eng.	60
98765	Bourikas	Elec. Eng.	98
98988	Tanaka	Biology	120

Neo4j

- Please set the values for your Neo4j database below.
- Make sure that your database is active. If you have not used it for a while, you need to log in through the website and restart the database.

In [4]:

```
neo4j_url = "neo4j+s://142430a7.databases.neo4j.io"
neo4j_user = "neo4j"
neo4j_password = 'plDygPT9TKfZxnMB9poFcUOBqlum7kw47-AHwFvg-uM'
```

In [5]:

```
from py2neo import Graph
```

In [6]:

```
def t1():
    graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
    q = "match (r:Person) where r.name='Tom Hanks' return r"
    res = graph.run(q)

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

- Please rerun the following cell.

In [7]:

```
t1()
```

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

MongoDB

- Please set your URL for MongoDB Atlas and make sure that your cluster is not suspended.

In [8]:

```
mongodb_url = "mongodb+srv://tedwu:W1fuq2peUOn5veXA@comp4111.rgzkcri.mongodb.net"
```

In [9]:

```
import pymongo
```

In [10]:

```
def connect():
    client = pymongo.MongoClient(
        mongodb_url
    )
    return client


def t_connect():
    c = connect()
    print("Databases = ", list(c.list_databases()))
```

In [11]:

```
#
# Note, you list of local databases will be different. The values do not matter.
#
t_connect()
```

```
Databases =  [{'name': 'hw4', 'sizeOnDisk': 835584, 'empty': False}, {'name': 'testd
b', 'sizeOnDisk': 40960, 'empty': False}, {'name': 'admin', 'sizeOnDisk': 380928, 'e
mpty': False}, {'name': 'local', 'sizeOnDisk': 11379404800, 'empty': False}]
```

Written Questions – General Knowledge

- The written questions require a short, succinct answer.
- Remember, "If you can't explain it simply, you don't understand it well enough."
- Some questions will research using the web, lecture slides, etc. You cannot cut and paste from sources. Your answer must show that you read the material and understand the concept.
- If you use a source other than lecture material, please provide a URL to the source(s) you read.

G1

Question: List at least two reasons why database systems support data manipulation using a declarative query language such as SQL, instead of just providing a library of C or C + + functions to carry out data manipulation.

Answer:

- A declarative query language allows the database system to optimize the execution of queries based on the available indexes, data distribution, and other factors. This can lead to improved performance compared to using a library of functions.
- A declarative query language allows users to focus on what data they want to retrieve or modify, rather than on how to retrieve or modify it. This separation of concerns makes it easier for users to write queries, as they do not have to worry about the underlying implementation details of the database.

G2

Question: List four significant differences between:

- Processing data by writing programs that manipulate files.
- Using a database management system and query language.

Answer:

- The program do not has data independence, which the data must be modified when the program is modified. While the database management system and query language has data independence, so it will be easier to modify sepeartely.
- The data integrity can be implemented in database management system and query language by using constraint on the query, but the program can not provide data integrity.
- The result comes from the program is stored in the single file, but the result from database management system is selected and showed from different table, which allows more complex query.
- The program need to edit the process of querying the result from the database, but on query language there is no need to care the process of querying.
- The program cannot provide the recover or backup if the file is defective, but the database management system can.

G3

Question: List five responsibilities (functionality provided) of a database-management system. For each responsibility, explain the potential problems that would occur with the functionality.

Answer:

- Backup and recover. When the data loss occured, the data will permentely loss if no backup and recover.
- Security. Provide the authorise method for users. User without authorisation will not be permitted to enter the database system.
- Data consistency. The data may not be the same if this function does not exist.
- Isolation. If it is not exist, the data can be edited by two people at the same time.
- Data integrity. The database will not be constrained if this function is not exist.

G4

Question: We all use SSOL to choose and register for classes. Another option would be to have a single Google sheet (shared spreadsheet) that we all use to register for classes. What are problems with using a shared spreadsheet?

Answer:

- The spreadsheet is shareable and everyone can edit, which can cause the potential defective.
- The spreadsheet may not be able to store large-scale data, if so the spreadsheet may takes long time to read.
- All the data in spreadsheet needs to be edited manually, which costs lots of time.

G5

Question: NoSQL databases have become increasingly popular for supporting applications. List 3 benefits of or reasons for using NoSQL databases versus SQL/relational databases. List 3 benefits of relational databases versus NoSQL databases.

Answer:

NoSQL databases versus SQL databases:

- NoSQL has flexible schema, which allows to change the database as the requirement changes.
- NoSQL is horizontally scaleable, which allows distributed storage and add servers to load balancing.
- Can visualise the database using graph.

SQL databases versus NoSQL databases:

- The SQL databases must follow the steps to validates and prevent the corruption.
- SQL databases provide isolation. Concurrent threads will not affect each other.
- SQL databse is vertical scaleable, which allows to mitigate to a larger server.

Relational Model

R1

Question: A column in a relation (table) has a *type.* Consider implementing a `date` as `CHAR(10)` in the format `YYYY-MM-DD`. The lecture material states that attributes (column values) come from a *domain.* Using `date` explain the differenc between a *domain* and a *type.*

Answer:

The domain and type can both define what should consist in a specific column. The domain can modify the allowed value, for example, the domain can be char(10), where char is the type. The type is the lowest level to define the column attribute.

R2

Question: The domain for a relation (table) attribute (column) should be *atomic.* Why?

Answer:

Since each attribute in a table should be the lowest unit and cannot be divided anymore, which helps to solve the case where some attributes are missing and can be recorded as NULL.

R3

Question: "In the US Postal System, a delivery point is a specific set of digits between 00 and 99 assigned to every address. When combined with the ZIP + 4 code, the delivery point provides a unique identifier for every deliverable address served by the United States Postal Service."

The lecture 2 slides provide a notation for representing a relation's schema. Assume we want to define a relation for US mailing addresses, and that the columns are:

- Zip code
- +4 code
- delivery_point
- address_line_1
- address_line_2
- city
- state

Use the notation to define the schema for an address. A simple example of an address's column values might be:

- Zip code: 10027
- +4 code: 6623
- delivery_point: 99
- address_line_1: 520 W 120th St
- address_line_2: Room 402
- city: New York

- state: NY

Answer:

mail_address(zip_code, four_code, delivery_point, address_line_1, address_line_2, city, state)

R4

Note: Use the RelaX (https://dbis-uibk.github.io/relax/calc/gist/4f7866c17624ca9dfa85ed2482078be8/relax-silberschatz-english.txt/0) calculator and the schema associated with the recommended textbook to answer this question. Your answer should contain:

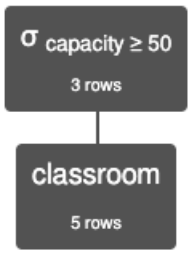
- The text for the query.
- An image showing the query execution and result.

An example of the format is:

Query

```
σ capacity >= 50 (classroom)
```

Execution



σ capacity ≥ 50 (classroom)

Execution time: 2 ms

classroom.building	classroom.room_number	classroom.capacity
'Packard'	101	500
'Taylor'	3128	70
'Watson'	120	50

‹ 1 ›

Question: Translate the following SQL statement into an equivalent relational algebra statement.

```
select
    *
from
    instructor
where
    dept_name in (select dept_name from department where budget >= 100000)
```

Answer:

π ID, name, instructor.dept_name, salary σ budget ≥ 100000 (instructor ⋈ instructor.dept_name = department.dept_name department)

R5

Use the same format to answer this question.

Question

Use the following query to compute a new table.

```
section_and_time =
    π course_id, sec_id, semester, year,
    day, start_hr, start_min (section ⋈ time_slot)
```

Using only section_and_time, write a relational algebra expression that returns a relation of overlapping courses of the form

```
(course_id_1, sec_id_1, semester_1, year_1, course_id_2, sec_id_2, semester_2, year_2) .
```

Your table cannot container duplicates. For example, a result containing

```
(BIO-101, 1, fall, 2022, MATH-101, 2, fall, 2022)
(MATH-101, 2, fall, 2022, BIO-101, 1, fall, 2022)
```

is incorrect.

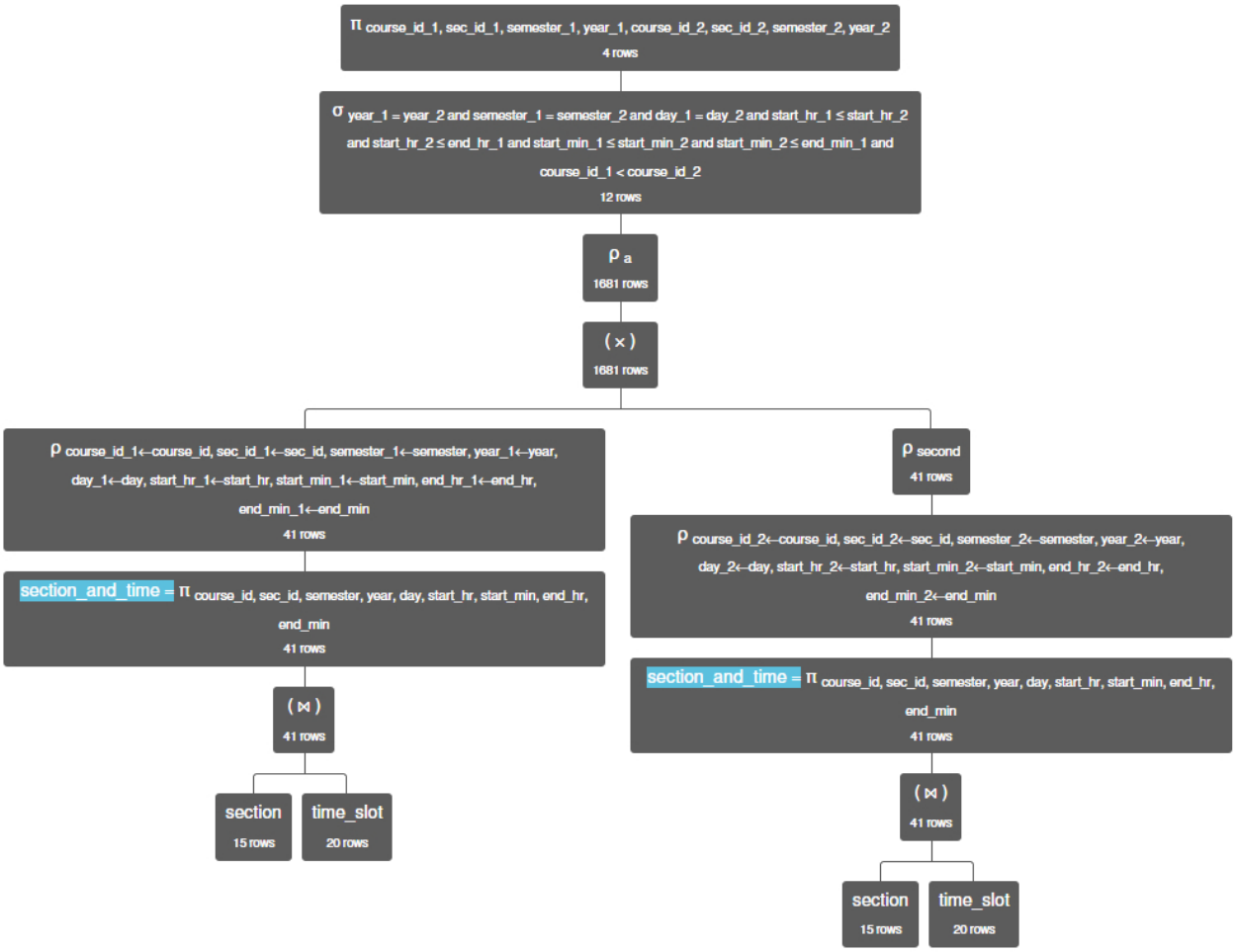
Answer:

Query

section_and_time = π course_id, sec_id, semester, year, day, start_hr, start_min, end_hr, end_min (section ⋈ time_slot)

π course_id_1, sec_id_1,semester_1, year_1, course_id_2, sec_id_2, semester_2, year_2 σ year_1 = year_2 and semester_1 = semester_2 and day_1 = day_2 and start_hr_1 ≤ start_hr_2 and start_hr_2 ≤ end_hr_1 and start_min_1 ≤ start_min_2 and start_min_2 ≤ end_min_1 and course_id_1 < course_id_2 (ρ a (ρ course_id_1←course_id, sec_id_1←sec_id, semester_1←semester, year_1←year, day_1←day, start_hr_1←start_hr, start_min_1←start_min, end_hr_1←end_hr, end_min_1←end_min section_and_time × ρ second (ρ course_id_2←course_id, sec_id_2←sec_id, semester_2←semester, year_2←year, day_2←day, start_hr_2←start_hr, start_min_2←start_min, end_hr_2←end_hr, end_min_2←end_min section_and_time)))

Execution



$$\pi_{\text{course\_id\_1, sec\_id\_1, semester\_1, year\_1, course\_id\_2, sec\_id\_2, semester\_2, year\_2}} \sigma_{\text{year\_1 = year\_2 and semester\_1 = semester\_2 and day\_1 = day\_2 and start\_hr\_1} \leq \text{start\_hr\_2 and start\_hr\_2} \leq \text{end\_hr\_1 and start\_min\_1} \leq \text{start\_min\_2 and start\_min\_2} \leq \text{end\_min\_1 and course\_id\_1} < \text{course\_id\_2} \left(\rho_{a} \left(\rho_{\text{course\_id\_1} \leftarrow \text{course\_id, sec\_id\_1} \leftarrow \text{sec\_id, semester\_1} \leftarrow \text{semester, year\_1} \leftarrow \text{year, day\_1} \leftarrow \text{day, start\_hr\_1} \leftarrow \text{start\_hr, start\_min\_1} \leftarrow \text{start\_min, end\_hr\_1} \leftarrow \text{end\_hr, end\_min\_1} \leftarrow \text{end\_min}} \pi_{\text{course\_id, sec\_id, semester, year, day, start\_hr, start\_min, end\_hr, end\_min}} \left(\text{section} \bowtie \text{time\_slot} \right) \times \rho_{\text{second}} \left(\rho_{\text{course\_id\_2} \leftarrow \text{course\_id, sec\_id\_2} \leftarrow \text{sec\_id, semester\_2} \leftarrow \text{semester, year\_2} \leftarrow \text{year, day\_2} \leftarrow \text{day, start\_hr\_2} \leftarrow \text{start\_hr, start\_min\_2} \leftarrow \text{start\_min, end\_hr\_2} \leftarrow \text{end\_hr, end\_min\_2} \leftarrow \text{end\_min}} \pi_{\text{course\_id, sec\_id, semester, year, day, start\_hr, start\_min, end\_hr, end\_min}} \left(\text{section} \bowtie \text{time\_slot} \right) \right) \right)$$

Execution time: 4 ms

a.course_id_1	a.sec_id_1	a.semester_1	a.year_1	a.course_id_2	a.sec_id_2	a.semester_2	a.year_2
'CS-315'	1	'Spring'	2010	'MU-199'	1	'Spring'	2010
'CS-319'	1	'Spring'	2010	'FIN-201'	1	'Spring'	2010
'CS-319'	2	'Spring'	2010	'HIS-351'	1	'Spring'	2010
'CS-347'	1	'Fall'	2009	'PHY-101'	1	'Fall'	2009

SQL

- You will use the Classic Models tutorial database (https://www.mysqltutorial.org/mysql-sample-database.aspx), which you should have already loaded into MySQL.

S1

Question: Create a view `employee_customer_sales` with the following information:

- `employeeNumber`
- `employeeLastname`
- `employeeFirstName`
- `customerNumber`
- `customerName`
- `revenue`
- The employee information is for the employee that is the `customer.customerRepEmployeeNumber`.
- `revenue` is the total revenue over all of the customer's orders.
  - The revenue for an `order` is `priceEach*quantityOrdered` for each `orderdetails` in the order.

Answer:

In [13]:

```
%%sql
use classicmodels;
CREATE VIEW employee_customer_sales AS
    SELECT
        c.salesRepEmployeeNumber AS 'employeeNumber',
        e.lastName AS 'employeeLastname',
        e.firstName AS 'employeeFirstname',
        tmp.customerNumber, c.customerName,
        sum(revenue) AS 'revenue'
    FROM
        (
        SELECT
            sum(orderdetails.quantityOrdered*orderdetails.priceEach) AS 'revenue',
            orderdetails.orderNumber,
            o.customerNumber
        FROM
            orderdetails
            JOIN
                orders o
                on orderdetails.orderNumber = o.orderNumber
        GROUP BY
            orderNumber
        ) tmp
        JOIN customers c
        ON tmp.customerNumber=c.customerNumber

        JOIN employees e
        ON c.salesRepEmployeeNumber = e.employeeNumber
    GROUP BY
        (tmp.customerNumber)
    ORDER BY
        employeeLastname, revenue DESC
```

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

Out[13]:

```
[]
```

Test Answer:

In [14]:

```
%sql select * from employee_customer_sales;
```

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

S2

Question:

- Below, there is a query that creates a view. Run the query.
- Using the view, write a query that produces a table of the form `(productCode, productName)` for products that no customer in Asia has ordered.
- For this questions purposes, the Asian countries are:
  - Japan
  - Singapore
  - Philipines
  - Hong King
- You must not use a JOIN.

In [15]:

```
#
# Create the view
#
%sql create or replace view orders_all as \
    select * from orders join orderdetails using(orderNumber)
```

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

Out[15]:

```
[]
```

Answer:

In [19]:

```
%%sql

DROP TABLE IF EXISTS final_s2;
CREATE TABLE final_s2 AS
    SELECT
        distinct(orders_all.productCode),
        products.productName
    FROM
        orders_all,
        products,
        customers
    WHERE
        orders_all.productCode = products.productCode
      AND
        orders_all.customerNumber=customers.customerNumber;
DELETE FROM
    final_s2
WHERE
    productCode
    IN
    (
    SELECT
        distinct(orders_all.productCode)
    FROM
        orders_all,
        products,
        customers
    WHERE
        orders_all.productCode = products.productCode
      AND
        orders_all.customerNumber=customers.customerNumber
      AND
        (
         customers.country = 'Japan'
             OR
         customers.country = 'Singapore'
             OR
         customers.country = 'Philippines'
             OR
         customers.country = 'Hong Kong'
        )
    );
select * FROM final_s2
```

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

Out[19]:

productCode	productName
S18_2870	1999 Indy 500 Monte Carlo SS
S24_2022	1938 Cadillac V-16 Presidential Limousine
S24_2972	1982 Lamborghini Diablo

S18_1342	1937 Lincoln Berline
S18_2795	1928 Mercedes-Benz SSK
S10_1678	1969 Harley Davidson Ultimate Chopper
S18_1367	1936 Mercedes-Benz 500K Special Roadster
S18_3320	1917 Maxwell Touring Car
S700_3505	The Titanic
S12_2823	2002 Suzuki XREO
S24_4258	1936 Chrysler Airflow
S10_4757	1972 Alfa Romeo GTA
S18_3029	1999 Yamaha Speed Boat
S18_3856	1941 Chevrolet Special Deluxe Cabriolet

S3

Question:

- Use the `customers` and `orders` for this query.
- Shipping days is the number of days between `orderDate` and `shippedDate`.
- Product a table of the form:
  - `customerNumber`
  - `customerName`
  - `noOfOrders` is the number of orders the customer placed.
  - `averageShippingDays` , which is the average shipping days.
  - `minimumShippingDays` , which is the minimum shipping days.
  - `maximumShippingDays` , which is the maximum shipping days.
- The table should only contain entries where:
  - `noOfOrders >= 3`
  - `averageShippingDays >= 5` or `maximumShippingDays >= 10`.

Answer:

In [20]:

```
%%sql

DROP TABLE IF EXISTS final_s3;
CREATE TABLE final_s3 AS
    SELECT *
    FROM
        (
            SELECT
                customers.customerNumber AS 'customerNumber',
                customers.customerName AS 'customerName',
                COUNT(orders.orderNumber) AS 'noOfOrders',
                avg(DATEDIFF(orders.shippedDate, orders.orderDate)) AS 'averageShippingDays',
                max(DATEDIFF(orders.shippedDate, orders.orderDate)) AS 'maximumShippingDays',
                min(DATEDIFF(orders.shippedDate, orders.orderDate)) AS 'minimumShippingDays'
            FROM
                customers,
                orders
            WHERE
                orders.customerNumber = customers.customerNumber
            GROUP BY
                customers.customerNumber
        ) as tmp
    WHERE
        tmp.noOfOrders >=3
      AND
        (
        averageShippingDays >= 5
          OR
        maximumShippingDays >= 10
        );
select * from final_s3;
```

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

Out[20]:

customerNumber	customerName	noOfOrders	averageShippingDays	maximumShippingD
363	Online Diecast Creations Co.	3	5.0000	
385	Cruz & Sons Co.	3	5.3333	
148	Dragon Souveniers, Ltd.	5	14.6000	
198	Auto-Moto Classics Inc.	3	5.6667	
161	Technics Stores Inc.	4	5.2500	
205	Toys4GrownUps.com	3	5.3333	
276	Anna's Decorations, Ltd	4	5.0000	
462	FunGiftIdeas.com	3	5.0000	
448	Scandinavian Gift Ideas	3	5.5000	

328	Tekni Collectables Inc.	3	5.0000
209	Mini Caravy	3	5.6667
398	Tokyo Collectables, Ltd	4	5.5000

Graph Database – Neo4j

- You will use your online/cloud Neo4j database for these problems.
- You must have loaded the Movie sample data.

N1

Question:

- The relationship `REVIEWED` connects a `Person` and `Movie`, and has the properties `rating` and `summary`.
- Write Python code using `py2neo` that produces the following table.

In [36]:

```
import pandas
```

Answer:

In [37]:

```
graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
q = "MATCH (b:Person)-[r:REVIEWED]->(u:Movie) return b.name, r.rating, r.summary, u.title"
res = graph.run(q).data()
df = pandas.DataFrame(res)
df.columns = ['reviewer_name', 'rating', 'rating_summary', 'movie_title']
df
```

Out[37]:

	reviewer_name	rating	rating_summary	movie_title
0	Jessica Thompson	92	You had me at Jerry	Jerry Maguire
1	James Thompson	100	The coolest football movie ever	The Replacements
2	Angela Scope	62	Pretty funny at times	The Replacements
3	Jessica Thompson	65	Silly, but fun	The Replacements
4	Jessica Thompson	45	Slapstick redeemed only by the Robin Williams ...	The Birdcage
5	Jessica Thompson	85	Dark, but compelling	Unforgiven
6	Jessica Thompson	95	An amazing journey	Cloud Atlas
7	Jessica Thompson	68	A solid romp	The Da Vinci Code
8	James Thompson	65	Fun, but a little far fetched	The Da Vinci Code

N2

Question:

- There are relationships `ACTED_IN` and `DIRECTED` between `Person` and `Movie`.
- Write Python code that produces the following table that shows people or both acted in and directed a movie.

In [38]:

```
graph = Graph(neo4j_url, auth=(neo4j_user, neo4j_password))
q = "MATCH (b:Movie)-[r:ACTED_IN]-(u:Person)-[r1:DIRECTED]-(b:Movie) return u.name,b.title"
res = graph.run(q).data()
df = pandas.DataFrame(res)
df.columns = ['name', 'title']
df
```

Out[38]:

	name	title
0	Tom Hanks	That Thing You Do
1	Clint Eastwood	Unforgiven
2	Danny DeVito	Hoffa

MongoDB

- Run the following code using your Atlas MongoDB.

In [39]:

```
import json

client = pymongo.MongoClient(
    mongodb_url
)

with open("./episodes.json") as e_file:
    episodes = json.load(e_file)
    episodes = episodes["episodes"]

for e in episodes:
    client['w4111_final']['episodes'].insert_one(e)
```

In [40]:

```
ratings_df = pandas.read_csv("./got_title_ratings.csv")
ratings_info = ratings_df[['tconst', 'averageRating', 'numVotes']]
r_dict = ratings_info.to_dict("records")

for r in r_dict:
    client['w4111_final']['ratings'].insert_one(r)
```

Question:

Write Python code that uses an aggregation pipeline and operations to produce the following table.

In [61]:

```
# Requires the PyMongo package.
# https://api.mongodb.com/python/current
#
# Write the query/aggregation that produces result
```

In [59]:

```
client = pymongo.MongoClient(
    mongodb_url
)
collection = client['w4111_final']['episodes']

pipeline = [
    {
        u"$project": {
            u"seasonNum": 1.0,
            u"episodeNum": 1.0,
            u"episodeLink": {
                u"$substr": [
                    u"$episodeLink",
                    7.0,
                    9.0
                ]
            },
            u"episodeTitle": 1.0,
            u"_id": 0.0
        }
    },
    {
        u"$lookup": {
            u"from": u"ratings",
            u"localField": u"episodeLink",
            u"foreignField": u"tconst",
            u"as": u"ratings"
        }
    },
    {
        u"$replaceRoot": {
            u"newRoot": {
                u"$mergeObjects": [
                    {
                        u"$arrayElemAt": [
                            u"$ratings",
                            0.0
                        ]
                    },
                    u"$$ROOT"
                ]
            }
        }
    },
    {
        u"$project": {
            u"seasonNum": 1.0,
            u"episodeNum": 1.0,
            u"episodeTitle": 1.0,
            u"episodeLink": 1.0,
            u"averageRating": 1.0,
            u"numVotes": 1.0,
            u"_id": 0.0
        }
    },
    {
        u"$replaceWith": {
            u"seasonNum": u"$seasonNum",
            u"episodeNum": u"$episodeNum",
```

```
            u"episodeTitle": u"$episodeTitle",
            u"episodeLink": u"$episodeLink",
            u"avgRating": u"$averageRating",
            u"numVotes": u"$numVotes"
        }
    }
]

cursor = collection.aggregate(
    pipeline,
    allowDiskUse = False
)
result = []
for doc in cursor:
    result.append(doc)
```

In [60]:

```
info_df = pandas.DataFrame(list(result))
info_df = info_df[['seasonNum', 'episodeNum', 'episodeLink', 'episodeTitle', 'avgRating', 'numVotes']]
info_df
```

Out[60]:

	seasonNum	episodeNum	episodeLink	episodeTitle	avgRating	numVotes
0	1	1	tt1480055	Winter Is Coming	8.9	48686
1	1	2	tt1668746	The Kingsroad	8.6	36837
2	1	3	tt1829962	Lord Snow	8.5	34863
3	1	4	tt1829963	Cripples, Bastards, and Broken Things	8.6	33136
4	1	5	tt1829964	The Wolf and the Lion	9.0	34436
...	...	...	...	...	...	...
68	8	2	tt6027908	A Knight of the Seven Kingdoms	7.9	130844
69	8	3	tt6027912	The Long Night	7.5	215995
70	8	4	tt6027914	The Last of the Starks	5.5	165067
71	8	5	tt6027916	The Bells	6.0	192449
72	8	6	tt6027920	The Iron Throne	4.0	248318

73 rows × 6 columns

Data Modeling and Schema Definition

- This is an exciting, interesting problem that involves:
  - Using Crow's Foot Notation
  - Relational approaches to implementing specialization, aggregation, quaternary relations, composite attributes and multi-valued attributes.
  - Foreign keys, check constraints and triggers.
- I did the answer and it took 3 hours to do all the work. My normal rule of thumb is that students require about 15 times as much time as I need to produce an answer.

- I giggled like the Riddler in Batman about how much fun we were going to have working on this question, and then the following happened.

- So, there will not be any data modeling question on the exam. Darn!

Module II Questions

- The questions require brief, written answers.

Q1

Question:

Briefly explain:

- Functional Dependency
- Lossy Decomposition
- Normalization

Answer:

Functional Dependency: For each relation R, if says that the attribute Y is functional dependent on X, then each X value can determine an unique value of Y.

Lossy Decomposition: For a relation R, if says that R is decomposed into sub relaitons R1, R2, ..., Rn. Then, if the same relation of R cannnot be get by join the R1 and R2, then it can be said as lossy decomposition.

Normalisation: Which can be said as a process of decomposing the relations into relations with fewer attributes, where each attributes has completed form, and the decomposition is lossless.

Q2

Question:

Briefly explain:

- Serializability
- Conflict Serializability
- Deadlock
- Cascading Abort
- Two Phase Locking

Answer:

Serialisability: The serialisability is a method to check if a serial of executions are able to maintaining the database consistency or not. It ensures the serial execution is equivalent to the serial execution.

Conflict Serializability: Conflict serialisability means that if a schedule, which can be transformed to a serial schedule by changing the non-comflicting operations.

Deadlock: In concurrent system, deadlock a situation in which two or more processes are unable to proceed because each is waiting for one of the others to do something. In DBMS, the process should be transaction.

Cascading abort: Cascading abort is a situation, where a transaction abortion forces another transaction abort in order to prevent the second transction read an uncommitted data.

Two phase locking: If a transaction performs a first unlock action, then it should not request any further locks. The phase before the locking point is growing phase, another is shrinking phase.

Q3

Question:

Briefly explain:

- Logical block addressing, CHS addressing
- RAID-0, RAID-1, RAID-5
- Fixed length records, variable length records.

Answer:

- Logical block addressing, CHS addressing:

Logical block addressing using number to locate the sector. The CHS can convert to logical block addressing by using the equation LBA=(C*h*+*H)*s+S-1. Where "s" is the number of secors per track, and "h" is the head number per cylinder.

CHS addressing use three parameters to locate the sector unit. At first, the cylinder parameter can let the head arms move to a specific radius on the disk. Then, head parameter can active a specific arm head. Finally, the header will locate to a sector when the disk is spiining.

- RAID-0, RAID-1, RAID-5:

RAID-0 makes two physical disks to one logical disk. RAID-0 can let the data lay flat on two disks, which makes R/W speed 2X faster than two seprated disk, and has full capacity of two disks. However, RAID-0 makes disks unreliable, since one of the RAID-0 disk defective will cause unrecoverable data loss.

RAID-1 makes two physical discks to one logical disk, one is main and another is backup. When the data is written on the first disk, the second disk will mirror the data. At this case, if one of disk in RAID-1 is defective, the data is safe since other diskes have full backup for a disk. However, the cost of the high reliability is that the capacity of disks in RAID-1 according to the smallest capacity of disks.

RAID-5 makes several disks to form one logical disk, each disk contains a part of data and a parity block, which can restore the data when one of disks is defective. RAID-5 improve utility from RAID-0, and improve the disk speed from RAID-1. However, RAID-5 can only tolerance one disk defective, and the disk speed will decrease a lot when a disk is defective because of the recalculating data.

- Fixed length records, variable length records:

Fixed length records stores each record with fixed length, these records are stored into disk blocks, note that cross-block storing may occured. If a record need to be deleted from file, then move all subsequent records forward, or move the last record to this location, or flag the space and link all free space on a free list.

Variable length record will be used if a database need to store multiple type of records on a file, a record type allow variable length (varchar) or record type includes repeating fields. Slotted Page Structure is one approach of the variable length record, which will release the space and delete the entity number, all records in the left side of the deleted record will be right shifted, and right side will be left shifted to ensure the free space is on the centre of the block.

Q4

Question:

Briefly explain:

- Clustered Index
- Sparse Index
- Covering Index

Answer:

Clustered index: The cluster index reorders the record in the table which are store in the disk physically, by according to the table and view based on the key values.

Sparse index: The sparse index records some of the values in file. A range of index columns store the same block address, which will be fetched when data is needed to be searched.

Covering index: The covering index is an index that includes all of the columns that are needed to satisfy a query. Covering index covers the query, meaning that the users can look up all of the required data from the index itself, without having to access the actual data rows in the underlying table.

Q5

Question:

Briefly explain:

- Equivalent queries
- Hash Join
- Materialization, Pipelining

Answer:

Equivalent queries: Equivalent queries means that queries can fetch same result form any illegal database.

Hash join: Hash join building hash tables from one or both of the joined relations. Then, probing the table in order to compare the has code for equality.

Materialisation: Materialisation is a process to create and set the view of the result of evaluated operation for the query. Materialisaion evalutes multiple operations of queries and store the result in the relations.

Pipelining: Pipelining is also a process to create and set the view of the result of evaluated operation for the query. Pipelining do not use temp relations to store the evaluated result, and need cache region to output. The pipelining can merge multiple operations into a pipeline to decrease the tmp file, by passing the result of the current operation to the next operation.