

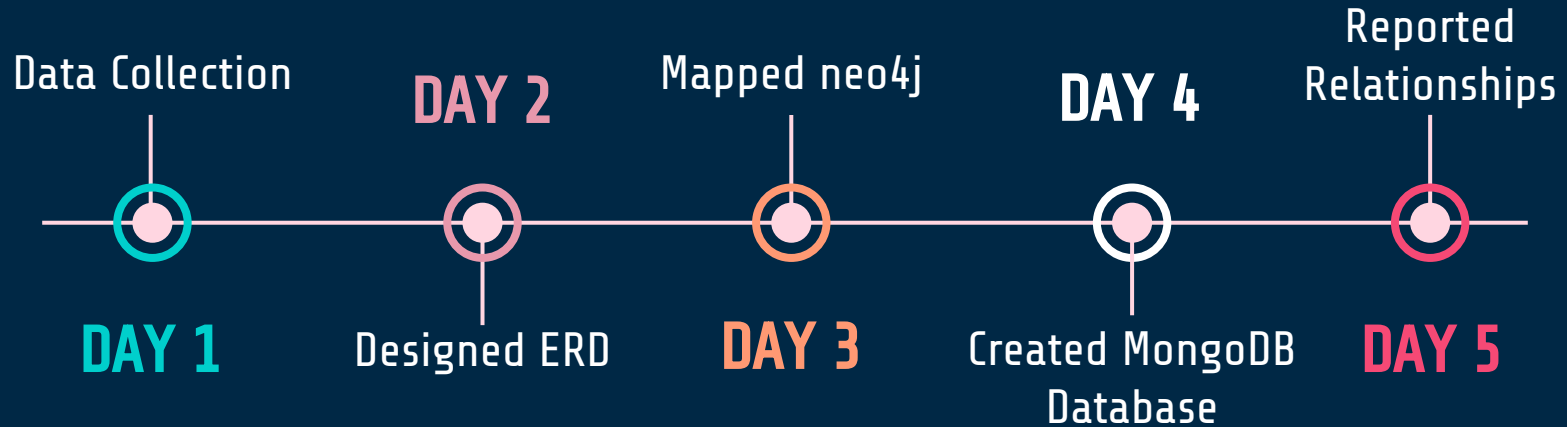
# WEEK 2 LAUNCH GROUP PROJECT

Drops of Jupyter  
Adi Pillai, Donald Cooper,  
Julia Duarte, Steven Burke,  
Sydney Ploeger

# Goals for Presentation

- Process
- Preliminary Visualizations
- Database Comparisons
  - Neo4j
  - Mongo
  - MySql
- Final Evaluation

# OUR PROCESS

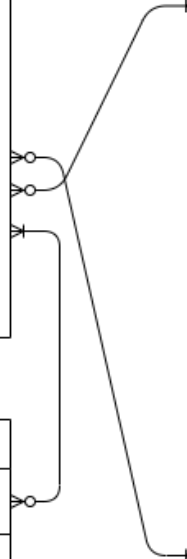


Names	
PK	<u>student_id str NOT NULL</u>
FK1	first str NOT NULL last str NOT NULL hometown str NOT NULL state str NOT NULL age int NOT NULL fav_class str NOT NULL fav_artist str NOT NULL fav_club str NOT NULL attends str NOT NULL forge_member bool NOT NULL

Music	
PK	<u>artist_name str NOT NULL</u>
FK1	genre str NOT NULL age_of_group int NOT NULL platinum_albums int NOT NULL most_pop_song str NOT NULL

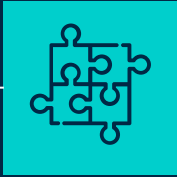
Clubs	
PK	<u>club_name str NOT NULL</u>
FK1	exec_member bool NOT NULL member_role str NOT NULL meeting_place(UVA building) str NULL tenure_terms(semesters) int NOT NULL active_member bool NOT NULL

Academics	
PK	<u>course_id str NOT NULL</u>
FK1	full_name str NOT NULL professor str NOT NULL dept_building str NOT NULL course_avg_rating int NOT NULL department str NOT NULL



# PRELIMINARY ERD

# DATABASE EXPLORATION



1

## NEO4J

Graphing visualizations of key-value relationships.



2

## MONGODB

Nesting based on shared relationships between tables.



3

## MYSQL

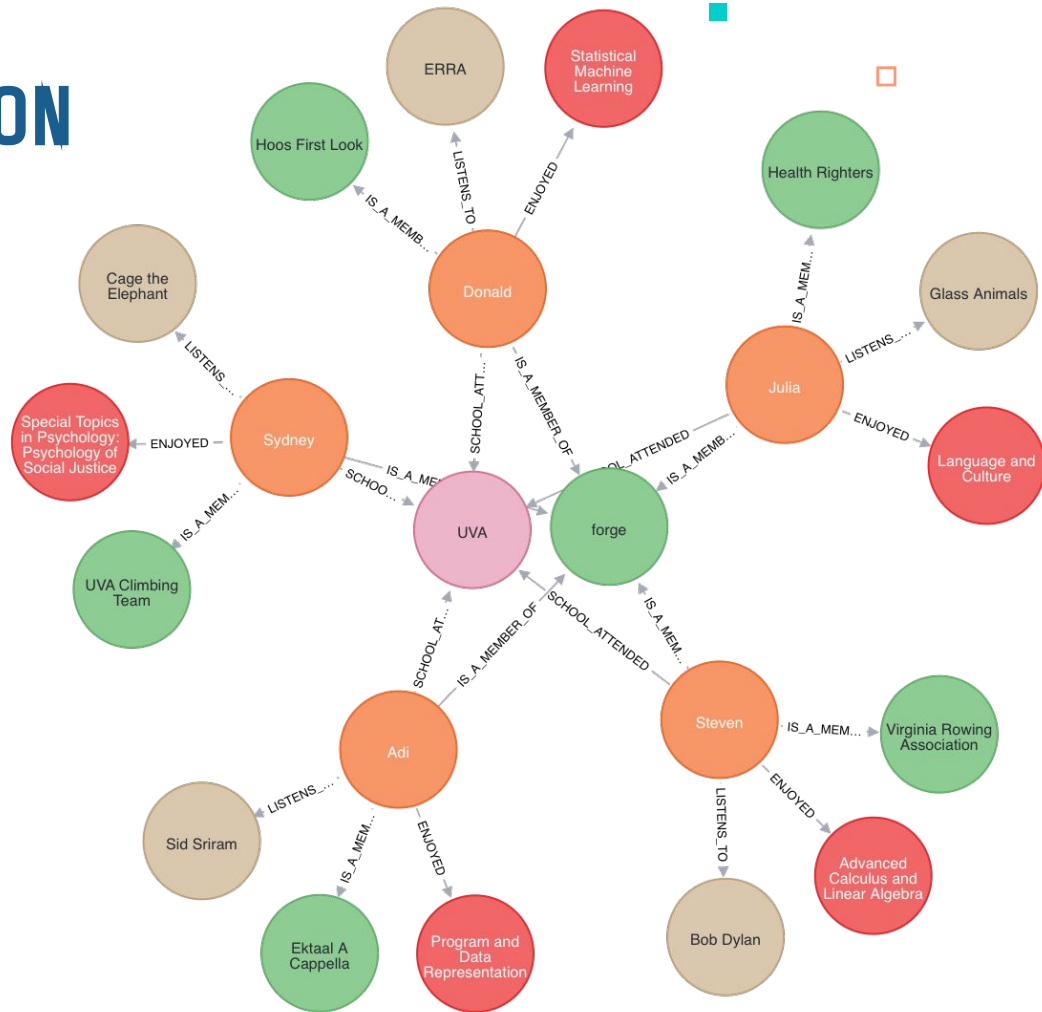
Querying unique relationships via RDBMS



NEO4J

1

# VISUALIZATION



# Advantages

- Easier visualization between connections.
- Performance during/after new data integration.
- Flexibility during industry changes.
- Agility prevents clunky development pipelines.

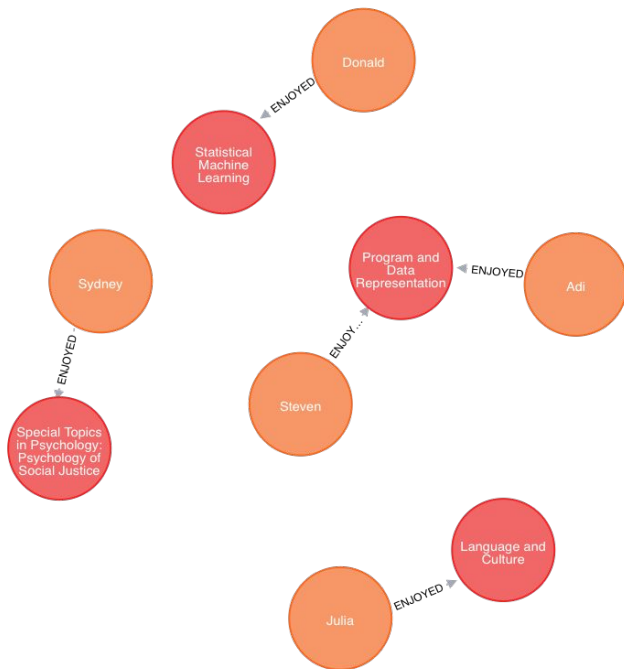


# Disadvantages

- Limited EDA and analytics hampers BI practices
- Not optimized for processing large datasets (“Bigdata”)
- Some relationships are difficult to map depending on database architecture
- More difficult aggregations

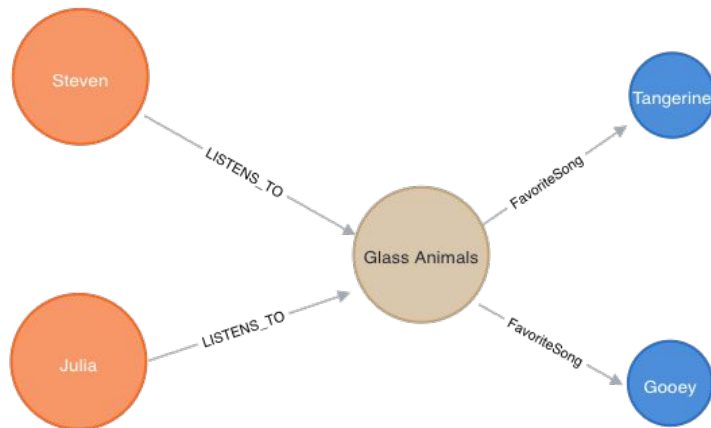
# Example Query 1

```
neo4j$ MATCH (a)-[:ENJOYED]→(b) return a,b
```



- Quickly visualizes all data points
- Can make comparisons for categorical data
- No bijective relationships

## Example Query 2



- Ownership of properties can be difficult
- Sometimes hard to map related but nuanced data

```
neo4j$ MATCH (a)-[:LISTENS_TO]→(b)-[:FavoriteSong]→(c) RETURN a,b,c
```

# MONGODB

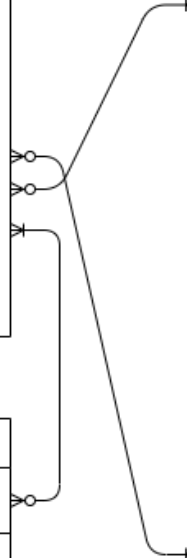
2

Names	
PK	<u>student_id str NOT NULL</u>
FK1	first str NOT NULL last str NOT NULL hometown str NOT NULL state str NOT NULL age int NOT NULL fav_class str NOT NULL fav_artist str NOT NULL fav_club str NOT NULL attends str NOT NULL forge_member bool NOT NULL

Music	
PK	<u>artist_name str NOT NULL</u>
FK1	genre str NOT NULL age_of_group int NOT NULL platinum_albums int NOT NULL most_pop_song str NOT NULL

Clubs	
PK	<u>club_name str NOT NULL</u>
FK1	exec_member bool NOT NULL member_role str NOT NULL meeting_place(UVA building) str NULL tenure_terms(semesters) int NOT NULL active_member bool NOT NULL

Academics	
PK	<u>course_id str NOT NULL</u>
FK1	full_name str NOT NULL professor str NOT NULL dept_building str NOT NULL course_avg_rating int NOT NULL department str NOT NULL



# PRELIMINARY ERD MONGODB

# VISUALIZATION

>

```
_id: ObjectId("60b8f42a5eae64752b92d79d")  
course_id: "CS 2150"  
full_name: "Program and Data Representation"  
department: "Computer Science "  
professor: "Aaron Bloomfield"  
dept_building: "Thornton Hall"  
course_avg_rating: 4.03
```

- CS 2150 Document from the Courses Collection
- Will be nested in our primary collection

# VISUALIZATION

## - Primary Collection

This document contains the previous document from the Courses Collection

```
>
  _id: ObjectId("60b8fa525eae64752b92d7ed")
  student_id: "arp3np"
  first: "Adi "
  last: "Pillai"
  hometown: "Moorestown "
  state: "NJ"
  age: 19
  fav_artist: Object
    _id: ObjectId("60b8f4f15eae64752b92d7ac")
    artist_name: "Sid Sriram"
    most_popular_song: "Inkem Inkem"
    genre: "R&B"
    age_of_group: 31
    platinum_albums: 0
  fav_class: Object
    _id: ObjectId("60b8f42a5eae64752b92d79d")
    course_id: "CS 2150"
    full_name: "Program and Data Representation"
    department: "Computer Science "
    professor: "Aaron Bloomfield"
    dept_building: "Thornton Hall"
    course_avg_rating: 4.03
  fav_club: Object
    _id: ObjectId("60b8f3ca5eae64752b92d794")
    club_name: "Ektaal A Cappella"
    exec_member: true
    member_role: "Treasurer"
    meeting_place: "AFC"
    tenure_terms: 4
    active_member: true
    attends: "UVA"
    forge_member: true
```

Example document from primary collection contains nested documents from sub-collections

# Advantages

- Easy to track all possible data for a given document
- Easily understood subdocuments
- Instituting new data is facilitated well by the document structure



# Disadvantages

- Sometimes difficult to compare data
- No standardization among the documents in a collection
- Learning curve not very intuitive

# Example Query



The screenshot shows a MongoDB query interface. At the top, there is a 'FILTER' button and a text input containing the query: `{age:{"$lt":20}, "$or":[{"fav_artist.age_of_group":{"$gt":20}},{"fav_artist.genre":"Alternative"}]}`. Below the input is a toolbar with an 'ADD DATA' button, a download icon, a 'VIEW' button, and three view mode icons (list, JSON, grid). The main area displays a document in JSON format:

```
_id: ObjectId("60ba450b1bfbe492e9bdc063")
student_id: "jod8esu"
first: "Julia"
last: "Duarte"
hometown: "Virginia Beach"
state: "VA"
age: 18
> fav_artist: Object
> fav_class: Object
> fav_club: Object
attends: "UVA"
forge_member: true
```

**Which students are younger than 20 and have a favorite artist older than 20?**

Able to filter on the fields of the document as well as the nested fields, which is unique to MongoDB

# Example Aggregation

```
1 ▾ /**
2   * _id: The id of the group.
3   * fieldN: The first field name.
4   */
5 ▾ {
6   _id: "$attends",
7   avg_rating: {"$avg" : "$fav_class.course_avg_ratin
8 }
```

```
_id: "UVA"
avg_rating: 4.266
```

**What is the average course rating for our favorite classes?**

Aggregations not as robust as in SQL or Node because the documents are independent from each other

Can still perform aggregations with documents in a collection if they have the same fields



# MYSQL

3

# VISUALIZATION

#	column_name	data_type	character_set	collation	is_nullable	column_default	extra	foreign_key
1	student_id	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	NO	NULL	EMPTY	EMPTY →
2	first	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY →
3	last	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY →
4	hometown	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY →
5	state	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY →
6	age	bigint	NULL	NULL	YES	NULL	EMPTY	EMPTY →
7	fav_artist	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	music(artist_name) →
8	fav_class	varchar(255)	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	academics(course_id) →
9	fav_club	bigint	NULL	NULL	YES	NULL	EMPTY	clubs(index) →
10	attends	text	utf8mb4	utf8mb4_0900_ai_ci	YES	NULL	EMPTY	EMPTY →
11	forge_member	tinyint(1)	NULL	NULL	YES	NULL	EMPTY	EMPTY →

The foreign key represents relationships between the "name" table's columns and another table's columns (i.e. "music", "academics", and "clubs").

index_name	index_algorithm	is_unique	column_name
fav_artist	BTREE	FALSE	fav_artist
fav_class	BTREE	FALSE	fav_class
fav_club	BTREE	FALSE	fav_club
PRIMARY	BTREE	TRUE	student_id

fav_artist	fav_class	fav_club
Sid Sriram →	CS 2150 →	4 →
ERRA →	STAT 4630 →	3 →
Glass Animals →	ANTH 2400 →	1 →
Bob Dylan →	MATH 2315 →	5 →
Cage the Elephant →	PSYC 4500 →	2 →

# Advantages

- Easily customizable queries
- More accessible comparisons
- Most familiar type of Database (relational databases)
- Can have a greater variety of data

# Disadvantages

- Queries and joins can be computationally demanding.
- Not as easy to visualize relationships within data.
- Stricter datatype requirements.

# Example Query 1

```
pd.read_sql("""SELECT * FROM names
             JOIN music ON names.fav_artist = music.artist_name
             JOIN academics ON names.fav_class = academics.course_id
             JOIN clubs ON names.fav_club = clubs.index
             WHERE age <= 20 AND hometown = 'Virginia Beach'""", con = cnx)
```

student_id	first	last	hometown	state	age	fav_artist	fav_class	fav_club	attends	...	dept_building
jod8esu	Julia	Duarte	Virginia Beach	VA	18	Glass Animals	ANTH 2400	1	UVA	...	Brooks Hall
sep2nb	Sydney	Ploeger	Virginia Beach	VA	20	Cage the Elephant	PSYC 4500	2	UVA	...	Gilmer Hall

Querying categorical information is simple with smaller data.

Which students are located in VA and are 20 years old or less?



## Example Query 2

```
pd.read_sql('''SELECT genre, platinum_albums,
                SUM(platinum_albums)
                FROM music
                GROUP BY genre
                ORDER BY SUM(platinum_albums) DESC;''', con = cnx)
```

	genre	platinum_albums	SUM(platinum_albums)
0	Folk	12	17.0
1	Rock	9	17.0
2	Pop	1	14.0
3	Progressive House	9	9.0
4	Rap	5	5.0
5	Electronic	3	3.0
6	Hip-Hop	3	3.0
7	Folk Rock	2	2.0

Which genre has the most platinum albums?

We can see the immediate advantages of query aggregations. However, we lose the visual appeal via table printouts.

# Best Choice of Database

Perfect for our rather  
small dataset

Best at visualizing  
relationships (students  
connections)

## Neo4j

Well suited for  
comparing categorical  
variables

Most visually appealing

# Other Considerations for Mongo and MySQL...

## MongoDB

- Nesting more information within existing columns.

## MySQL

- Calling more BI/Analytics queries for data summaries and/or aggregation.

# Additional Things!

Sydney Ploeger · Just now

## All About Databases

*Exploring Categories of Databases, Advantages and Disadvantages of Different Databases, and More...*

Adi Pillai, Donald Cooper, Julia Duarte, Steven Burke, & Sydney Ploeger

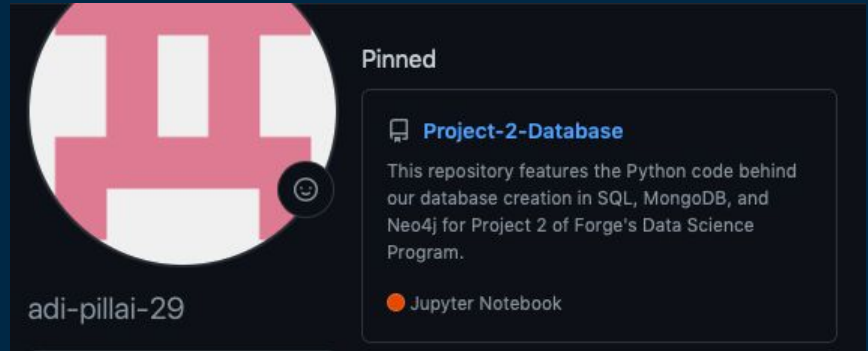
In the following, we will explain the general concept of databases, along with the technical aspects of these databases in regards to our own Week 2 project.

## Medium Article

<https://sep2nb.medium.com/all-about-databases-ca47cb3b784e>

## GitHub Repository

<https://github.com/adi-pillai-29/Project-2-Database.git>



The background is a dark blue gradient. It is decorated with a pattern of small squares in teal, orange, and pink, and thin white vertical lines of varying heights, creating a modern, geometric aesthetic.

# Thank you!

## Questions?