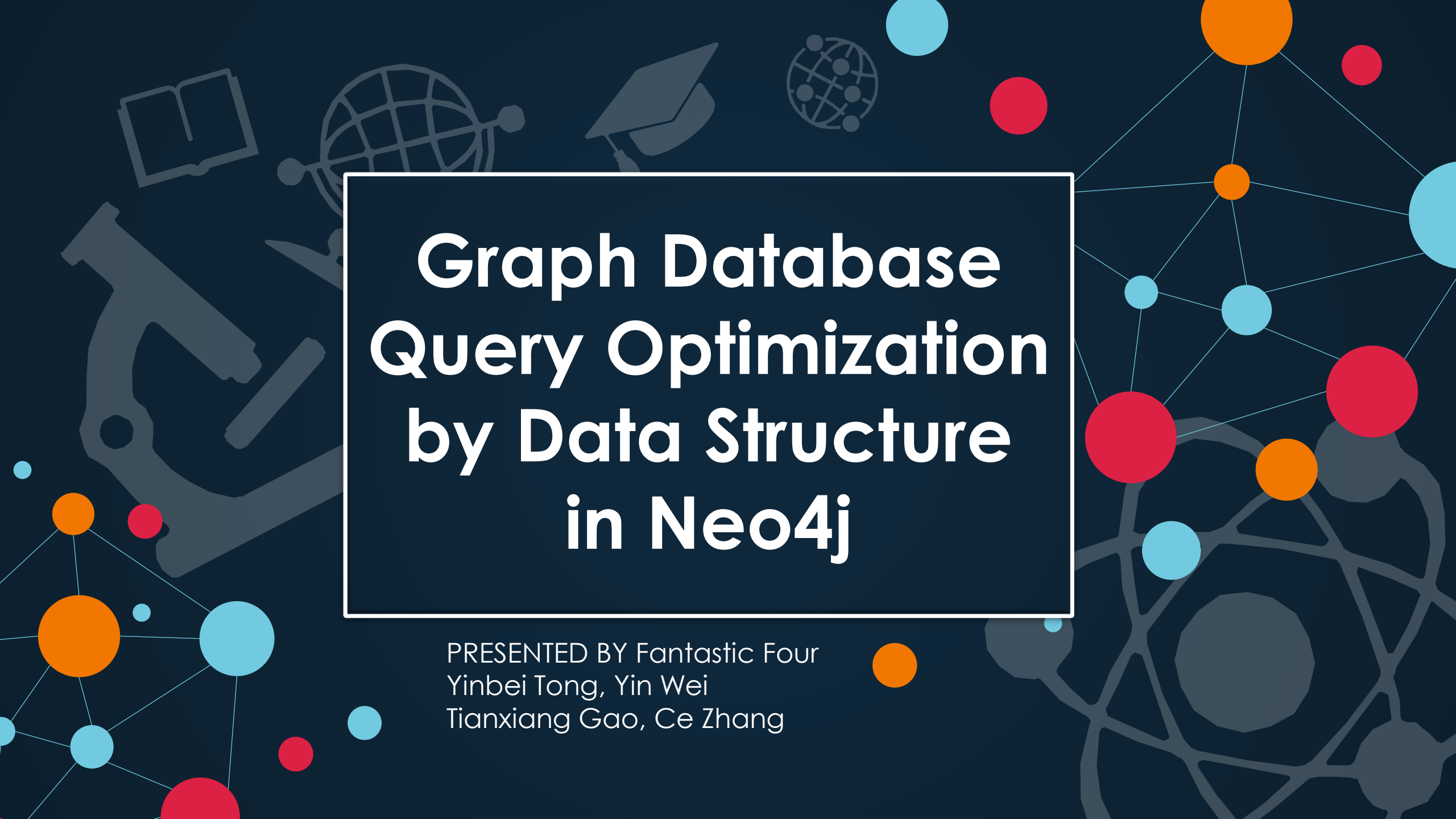


Graph Database Query Optimization by Data Structure in Neo4j

PRESENTED BY Fantastic Four
Yinbei Tong, Yin Wei
Tianxiang Gao, Ce Zhang

Part ONE

01 Brief Introduction

02 Related Work

03 Proposed Work

04 Data Collection

05 Data Analysis

06 Conclusion

PART TWO

Related Work

Ying Wei

Related Work

First paper: Dominguez-Sal, D., P. Urbón-Bayes, A. Giménez-Vañó, S. Gómez-Villamor, N. Martínez-Bazán, and J. L. Larriba-Pey. "**Survey of Graph Database Performance on the HPC Scalable Graph Analysis Benchmark.**" Web-Age Information Management Lecture Notes in Computer Science: 37-48. 2010.

Second paper: Bonnici, Vincenzo, Alfredo Ferro, Rosalba Giugno, Alfredo Pulvirenti, and Dennis Shasha. "**Enhancing Graph Database Indexing by Suffix Tree Structure.**" Pattern Recognition in Bioinformatics Lecture Notes in Computer Science: 195-203. 2010.

Third paper: Ciglan, Marek, Alex Averbuch, and Ladialav Hluchy. "**Benchmarking Traversal Operations over Graph Databases.**" 2012 IEEE 28th International Conference on Data Engineering Workshops.

The Second Paper

Problem: The application domains such as Bioinformatics and cheminformatics represent data as graph where nodes are basic element and edges model the relations among them. And the dataset are large and growing in size. Need to deal with graph searching efficiently.

Main idea: GraphGrepSX. The efficient graph searching algorithms together with filtering technique.

Continue

Preprocessing phase.

Filtering and matching.

Comparison: GraphGrepSX with SING, GraphFind, CTree and GCoding.

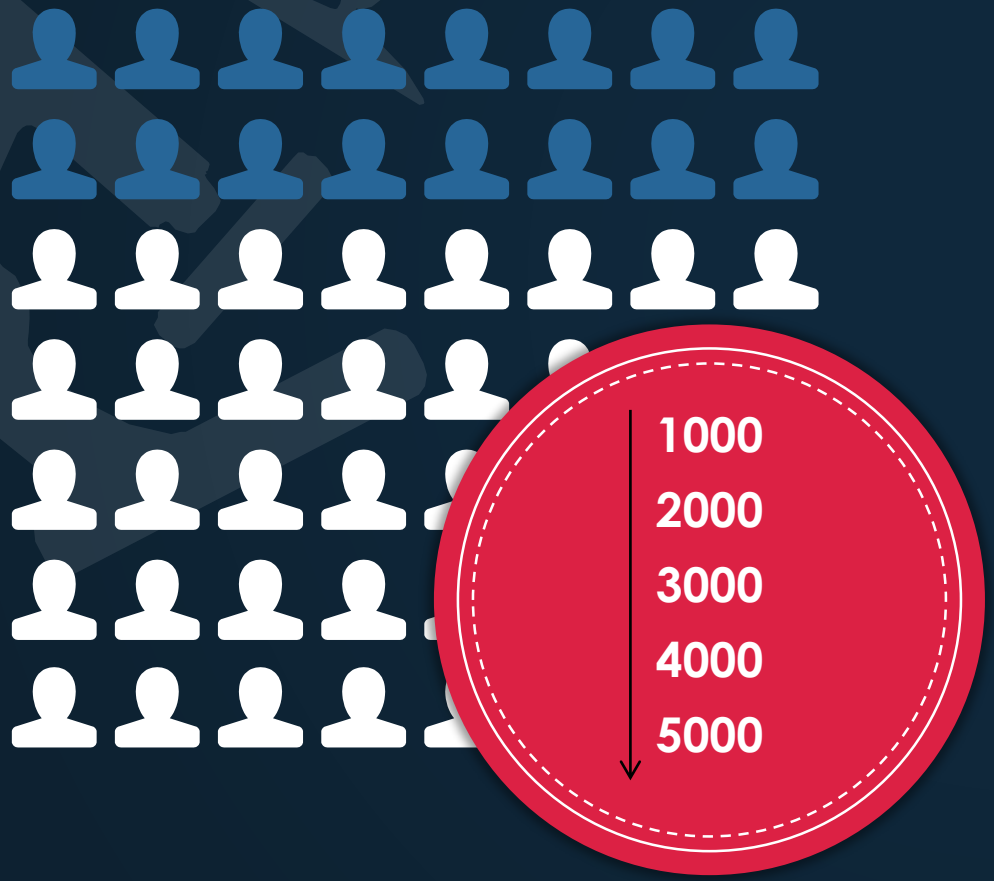
Conclusion: traversal path require less filtering and querying time.

PART THREE

Proposed Work

Tianxiang Gao

PART THREE Proposed Work



Dataset

- **5** Datasets
- Each Dataset has a number of person nodes
- Performed experiment on 5 datasets with following number of person nodes:
  1000, 2000, 3000, 4000, 5000
- Collected the construction time of 10000.
  Because the construction time is longer than
  20 mins when dataset larger than **5000**

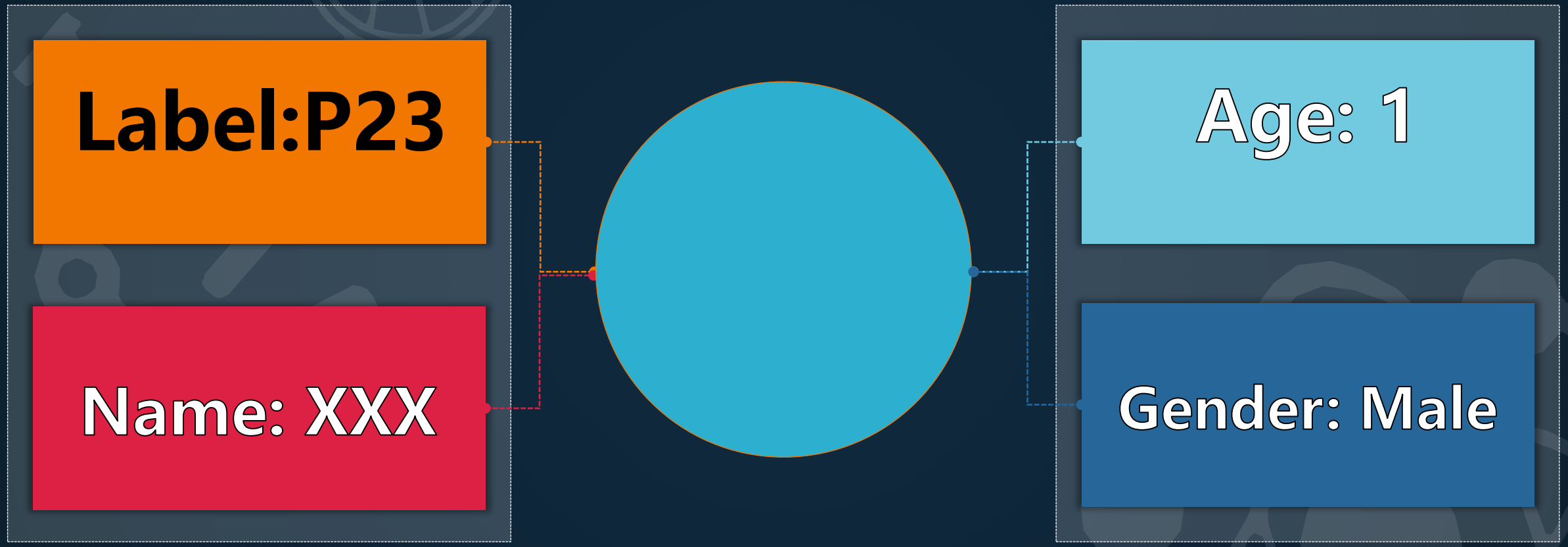
PART THREE Proposed Work

	D1	D2	D3	D4	D5	D6
1000	1000D1	1000D2	1000D3	1000D4	1000D5	1000D6
2000	2000D1	2000D2	2000D3	2000D4	2000D5	2000D6
3000	3000D1	3000D2	3000D3	3000D4	3000D5	3000D6
4000	4000D1	4000D2	4000D3	4000D4	4000D5	4000D6
5000	5000D1	5000D2	5000D3	5000D4	5000D5	5000D6

30 Databases

Each database we designed 37 number of queries to test the query performance

Person Node



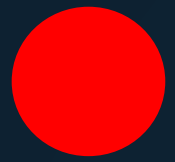
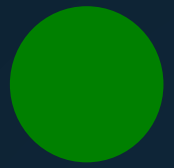
Age Node

$$i \in \{ 1 , \dots , 99 \}$$

Gender Node

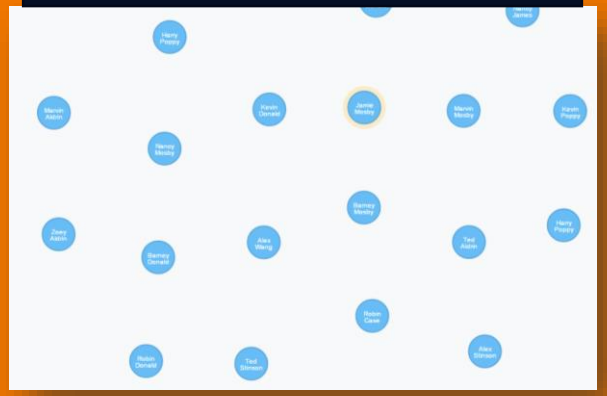
gender:
Male

gender:
Female

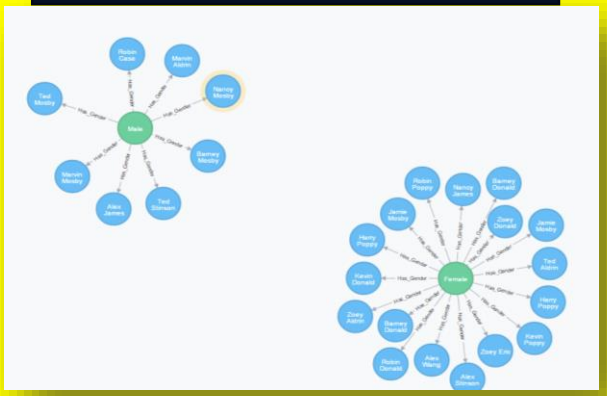
age:  gender:  Person: 

Data Structure

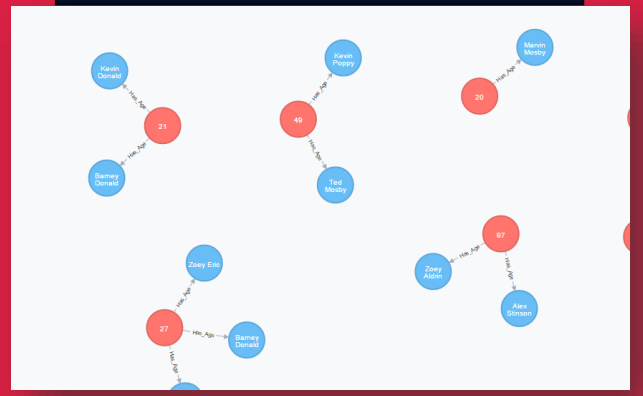
D1: only person nodes

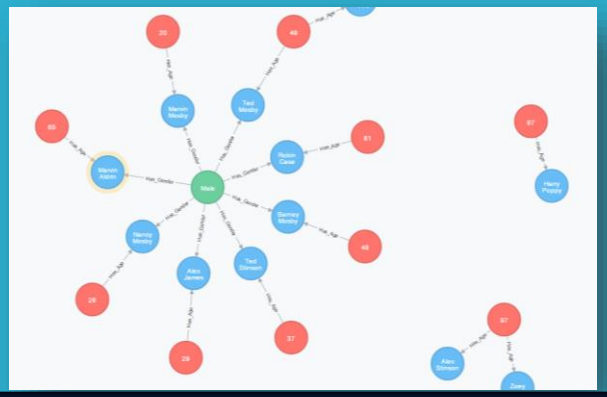


D2: gender->person

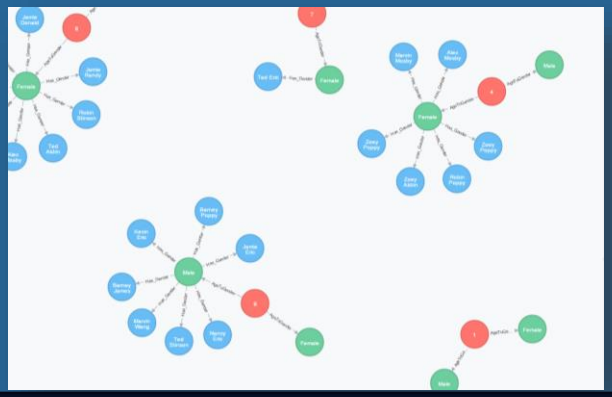


D3: age->person

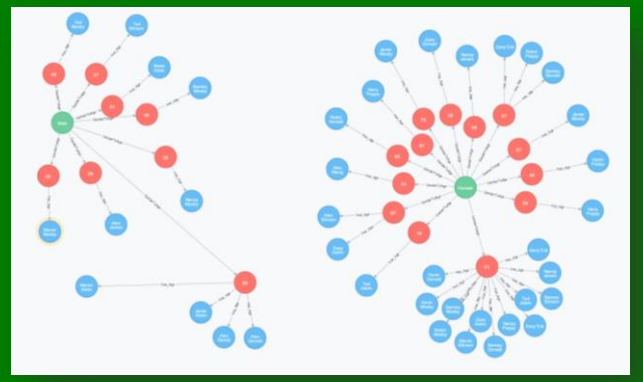




D4: gender-> person <-age



D5: age-> gender-> person



D6: gender-> age -> person

PART THREE Query Design

Query 1

No.	Query Properties	Selection Factor
	age	
1	0 - 9	10 %
2	0 - 19	20 %
3	0 - 29	30 %
4	0 - 39	40 %
5	0 - 49	50 %
6	0 - 59	60 %
7	0 - 69	70 %
8	0 - 79	80 %
9	0 - 89	90 %
10	0 - 99	100 %

Query 1

10% - 100%

Only by age

This query only query for age in a range.

Age range : 0 – 99 (Integer)

PART THREE Query Design

Query 2

No.	Query Properties	Selection Factor
	gender	
1	Male	50 %
2	Male & Female	100 %

Query 2

50% & 100%

Only by gender

- By Male
- Or by Male & Female
- Only by Male will have the same result as only by Female

PART THREE Query Design

Query 3

No.	Query Properties		Selection Factor
	gender	age	
1	Male	0 - 9	5 %
2	Male	0 - 19	10 %
3	Male	0 - 29	15 %
4	Male	0 - 39	20 %
5	Male	0 - 49	25 %
6	Male	0 - 59	30 %
7	Male	0 - 69	35 %
8	Male	0 - 79	40 %
9	Male	0 - 89	45 %
10	Male	0 - 99	50 %

Query 3

5% - 50%

By Male & age range

- Same result as by female & age range

PART THREE Query Design

Query 4

No.	Query Properties		Selection Factor
	gender=Male age range	gender=Female age range	
1	0 - 19	0 - 19	20 %
2	0 - 19	0 - 39	30 %
3	0 - 19	0 - 59	40 %
4	0 - 19	0 - 79	50 %
5	0 - 19	0 - 99	60 %
6	0 - 39	0 - 39	40 %
7	0 - 39	0 - 59	50 %
8	0 - 39	0 - 79	60 %
9	0 - 39	0 - 99	70 %
10	0 - 59	0 - 59	60 %
11	0 - 59	0 - 79	70 %
12	0 - 59	0 - 99	80 %
13	0 - 79	0 - 79	80 %
14	0 - 79	0 - 99	90 %
15	0 - 99	0 - 99	100 %

Query 4

20% - 100%

By mixed gender & age range

- Query matrix.
- Upper triangular matrix
- Male age(0-19) & Female age(0-39) will have the same query time as Female age(0-19) % Male age(0-39)

Unique Contribution

1. Complete test of query performance on the case when extract out two attributes with huge value number difference(2 and 100)

2. Test result can be the reference for Neo4j's future index method optimization to avoid designing certain index method in certain case.

PART FOUR

Data Collection

Ce Zhang

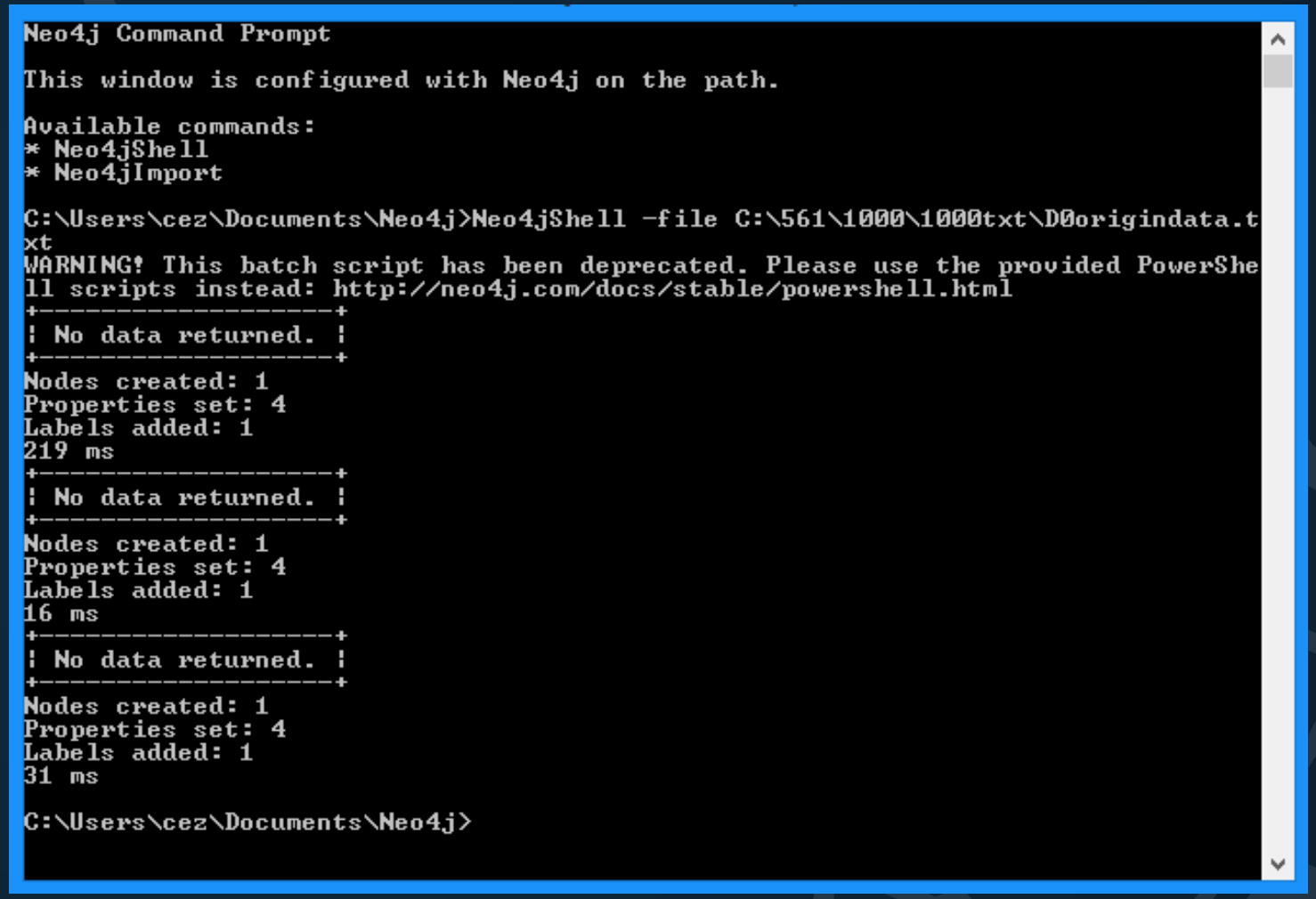
1. Run Java program to generate 30 database directories and 30 cypher.txt files.

2. Create database using command line Neo4jShell –file cypher.txt.

3. Run queries in each database to record time.

PART FOUR Data Collection

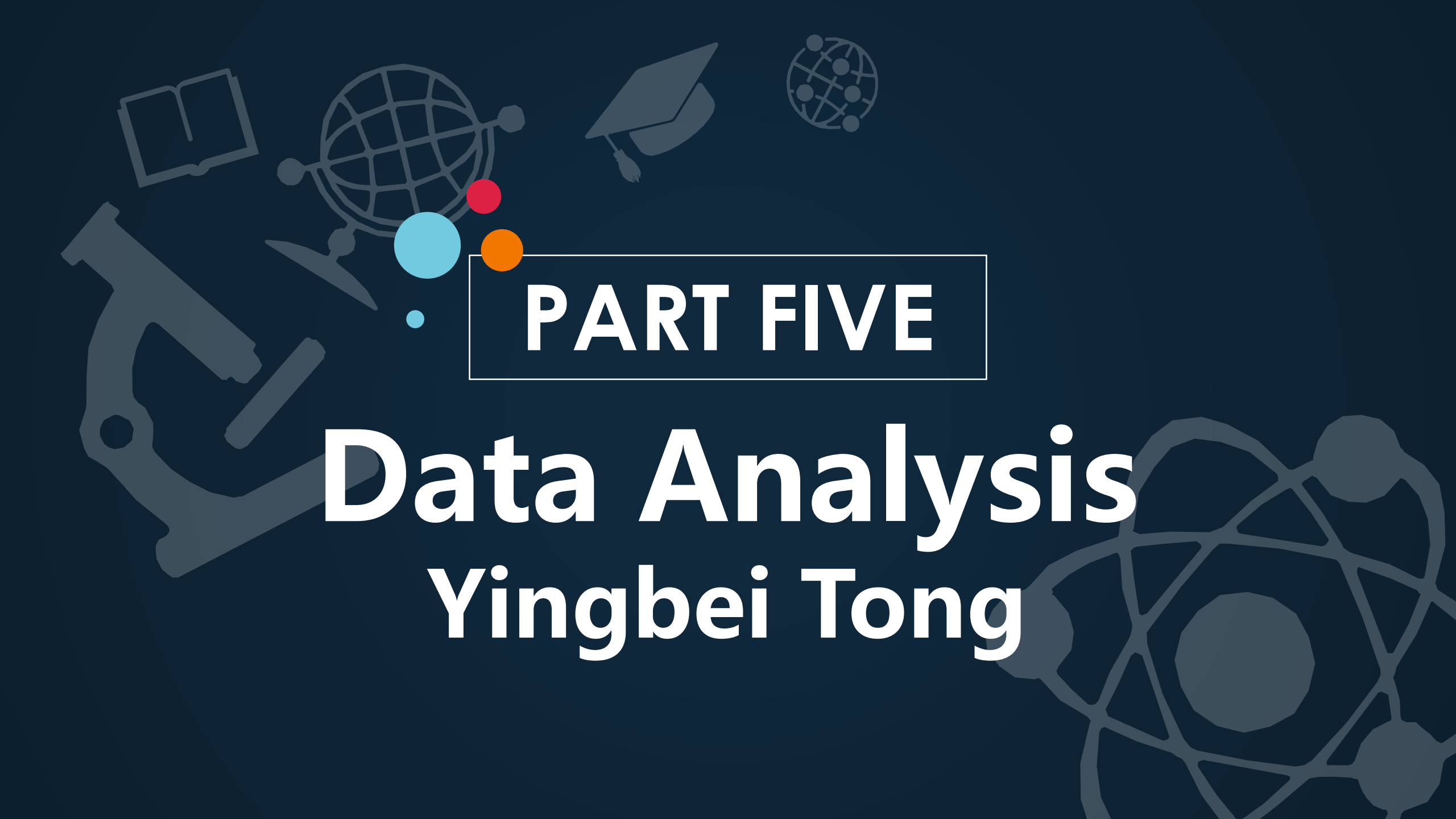
1. Record time of establishing each database.



PART FOUR Data Collection

2. Record time of each query in Neo4jShell.

```
neo4j-sh (?)$ match (p:Person) where p.age>=0 and p.age<2 return p;
+-------------------------------------------------------------------+
| p                                                                 |
+-------------------------------------------------------------------+
| Node[28]{gender:"Female",ID:29,name:"Marvin Wang",age:0}          |
| Node[158]{gender:"Female",ID:159,name:"Marvin Stinson",age:1}     |
| Node[186]{gender:"Female",ID:187,name:"Jamie Donald",age:1}       |
| Node[208]{gender:"Female",ID:209,name:"Marvin James",age:0}       |
| Node[310]{gender:"Female",ID:311,name:"Zoey Eric",age:0}          |
| Node[342]{gender:"Male",ID:343,name:"Robin Mosby",age:0}          |
| Node[390]{gender:"Male",ID:391,name:"Zoey Mosby",age:1}           |
| Node[487]{gender:"Male",ID:488,name:"Kevin Donald",age:0}         |
| Node[697]{gender:"Female",ID:698,name:"Kevin Eric",age:1}         |
| Node[710]{gender:"Female",ID:711,name:"Zoey Aldrin",age:1}        |
| Node[778]{gender:"Male",ID:779,name:"Marvin Donald",age:0}        |
| Node[851]{gender:"Male",ID:852,name:"Barney Donald",age:1}        |
| Node[853]{gender:"Female",ID:854,name:"Alex Poppy",age:0}         |
| Node[882]{gender:"Male",ID:883,name:"Robin Donald",age:0}         |
| Node[888]{gender:"Female",ID:889,name:"Marvin Wang",age:1}        |
| Node[954]{gender:"Male",ID:955,name:"Ted James",age:0}            |
| Node[960]{gender:"Female",ID:961,name:"Zoey Poppy",age:0}         |
| Node[987]{gender:"Male",ID:988,name:"Kevin James",age:1}          |
+-------------------------------------------------------------------+
18 rows
94 ms
```

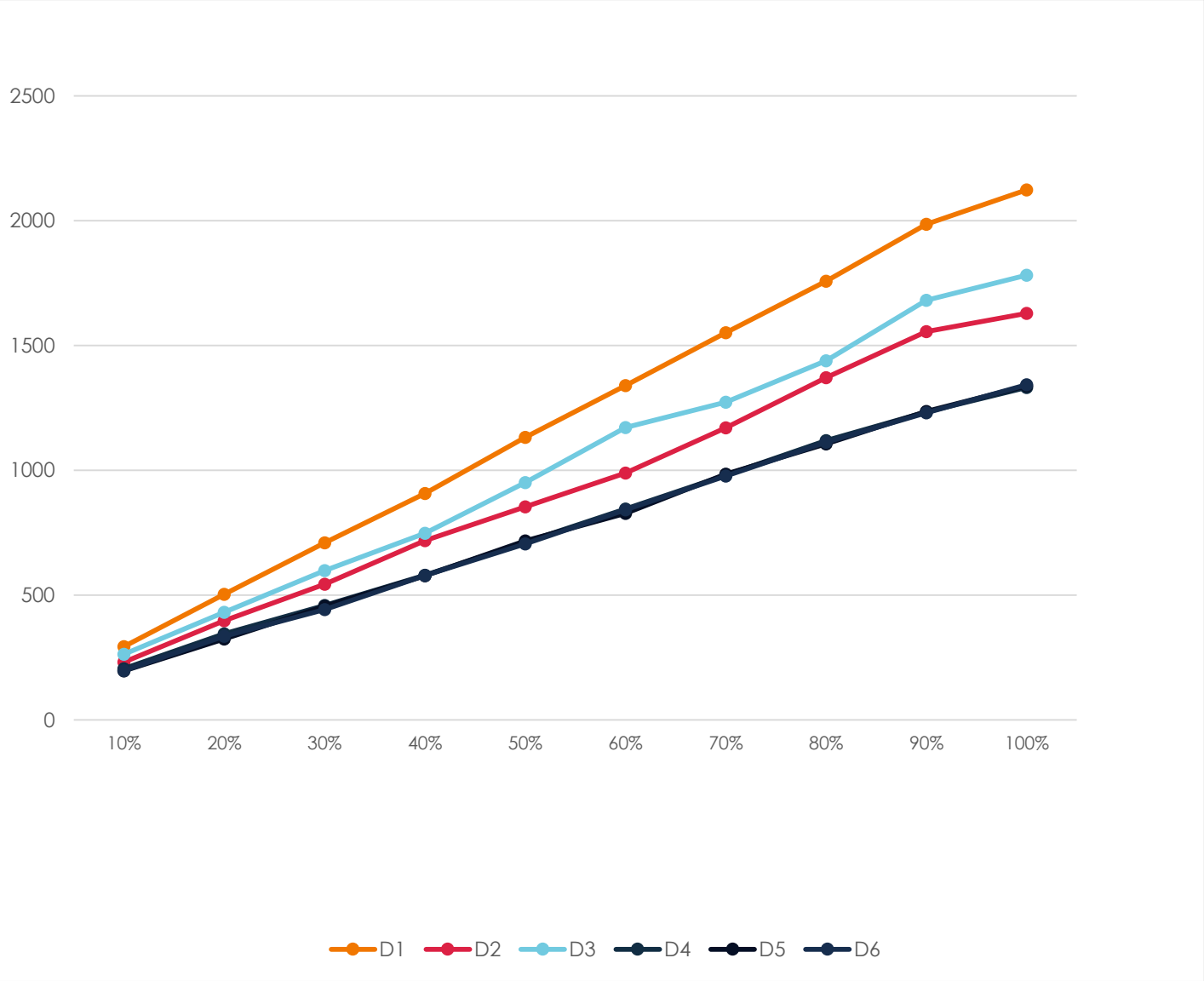


PART FIVE

Data Analysis

Yingbei Tong

PART 5 Data Analysis



Experiment Result

Figure 1: Query 1 on dataset 4 with different selection factor.

What we find are same with the third graph, and the gap between each line in this graph also bigger than that from previous three graphs.

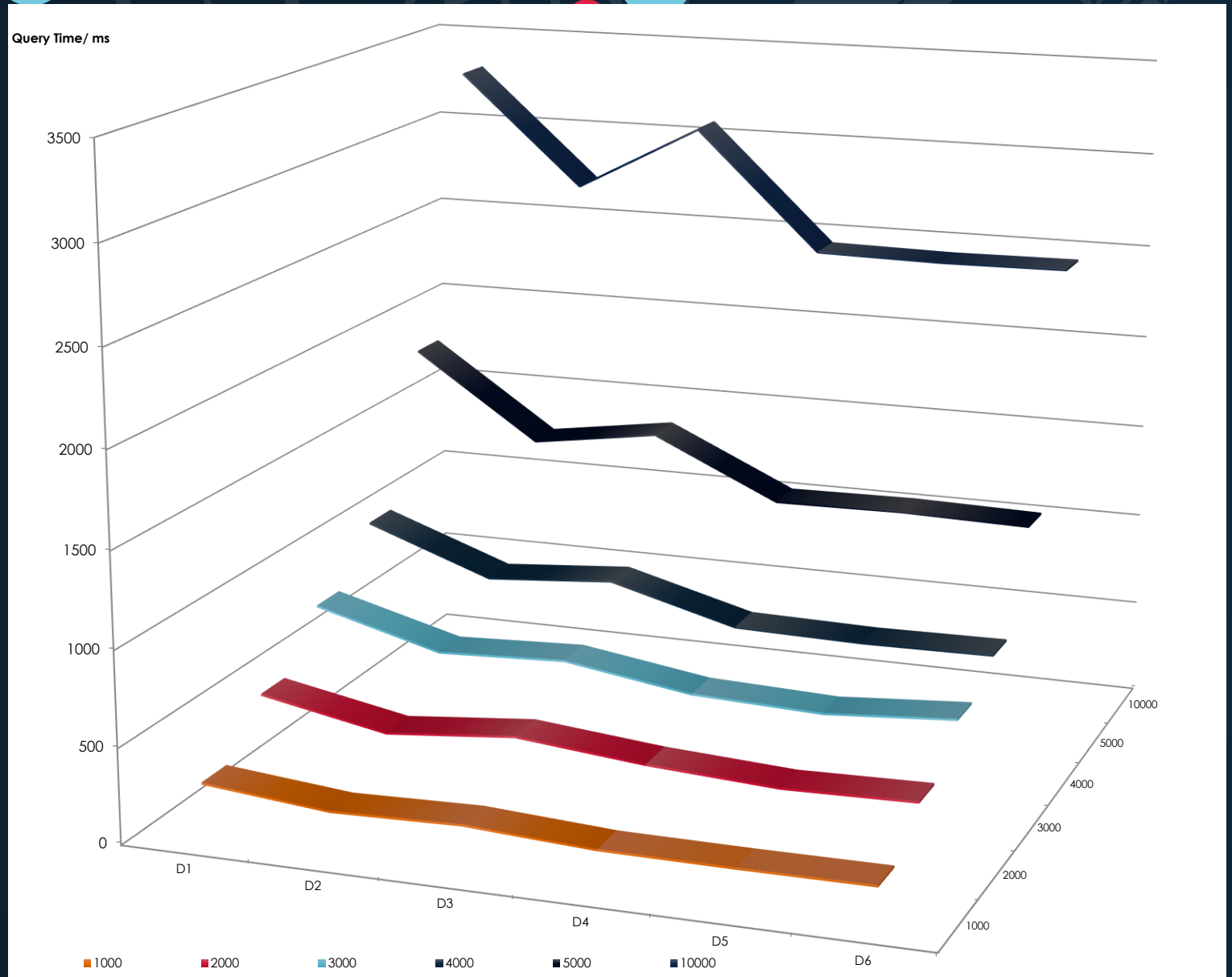
PART 5 Data Analysis

Experiment Result

Figure 2: Query 3 on dataset 4 with different selection Factor.

The query time trend still maintain its linear property. All trends and finds we get from this graphs is the same with the previous figure.

PART 5 Data Analysis

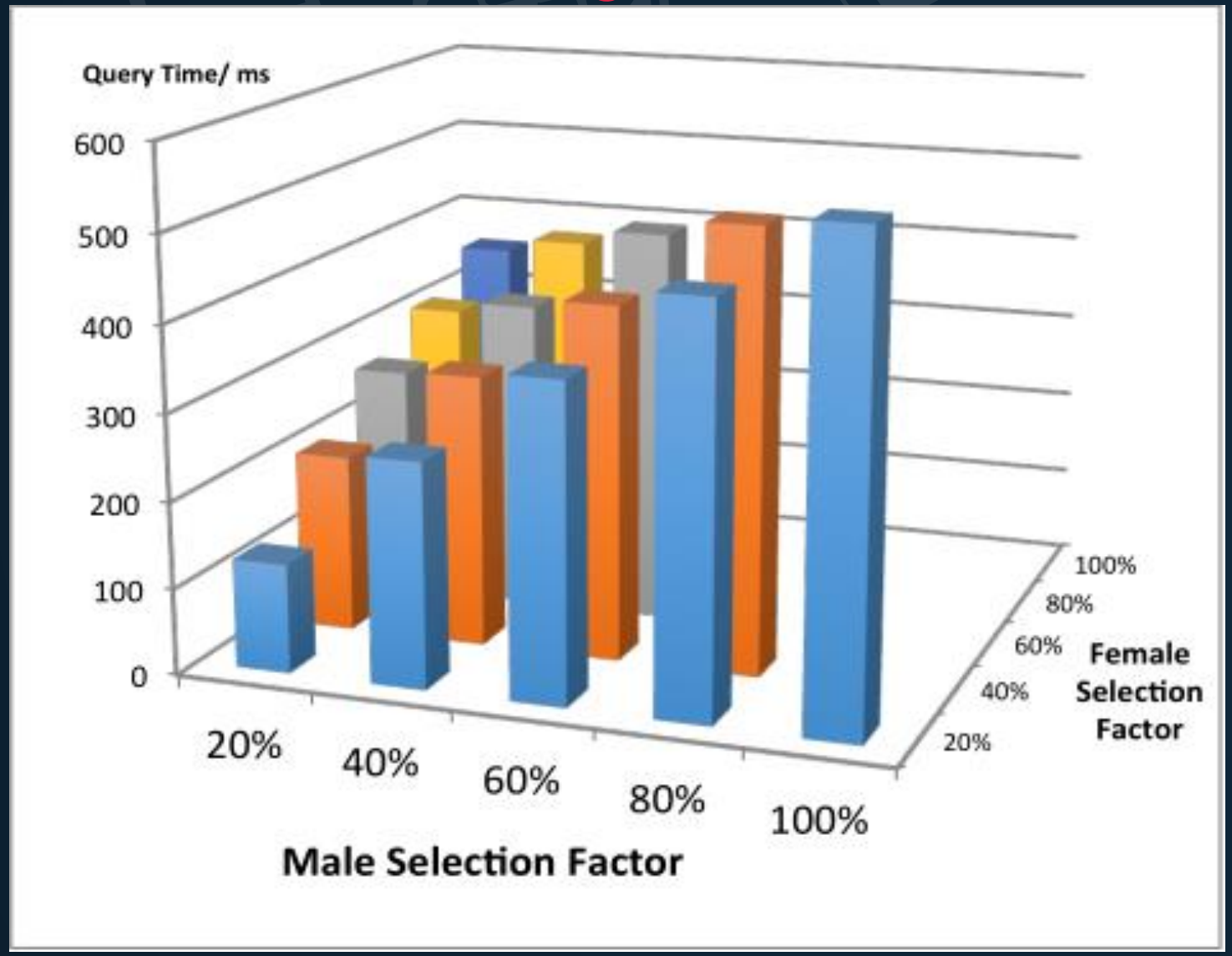


Experiment Result

Figure: querying all the Male People on different dataset With selection factor = 50%

show the increasing trend when the dataset size increasing. This figure displays the result that the databases have worse performance on third data structure (D3).

PART 5 Data Analysis

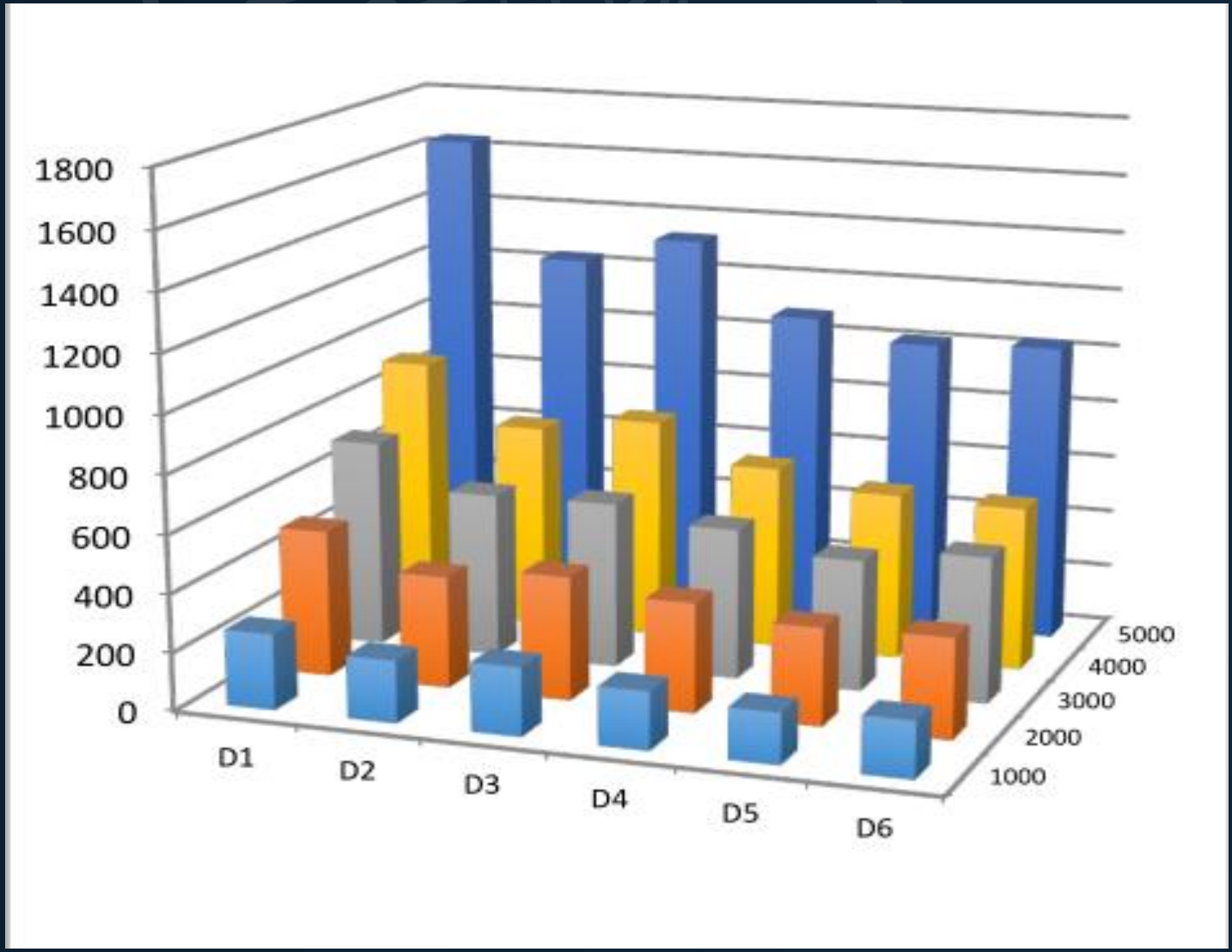


Experiment Result

Figure 4: shows when query both gender and age, how the original graph performance.

When mixed query both gender and age, the trends are still increasing with increasing selection factors.

PART 5 Data Analysis



Experiment Result

Figure 5 : shows the query time on data structure from D1 to D6.
We can see that the third data structure also shows a drawback on mix query properties.
We also see that on larger dataset, D5 and D6 have almost same performance level.

PART SIX Conclusion

As the selection factor grows linearly, the average response time also increases proportionally. Similarly, if the size of dataset grows linearly, the average response querying time increases linearly as well.

Those structures using popular attributes as index performed obviously better than the original structure. The structure (database 5 and 6) with 2-layer index returns lowest average response time thus has best performance.

When the selection factor reaches nearly 100 %, the data structure 5 and 6 will have much more edges than the original discrete nodes. However, query 5 and 6 still maintain a better performance than the original data structure.

When query for only one attributes, using both of two attributes to construct database will also return a better result than only extract one kind of attribute.

Drawback

1. Doesn't compare with other index method by our experiment benchmark
2. Doesn't compare with Neo4j's internal index method
3. Larger dataset size
4. Not sharp difference by this selection factors


Q & A