CHIGREECE 2023
CHAPTER
Connect. Engage. Inspire.

Human factors in software development

A study on database system adoption by developers

Ioanna Terzi
Human Computer Interaction Group,
University of Patras
ioanna346terzi@gmail.com

Monica Divitini
Department of Information and Computer Science,
Norwegian University of Science and Technology
divitini@ntnu.no

Nikolaos Avouris
Human Computer Interaction Group,
University of Patras
avouris@upatras.gr

CHIGREECE 2023
CHAPTER
Connect. Engage. Inspire.

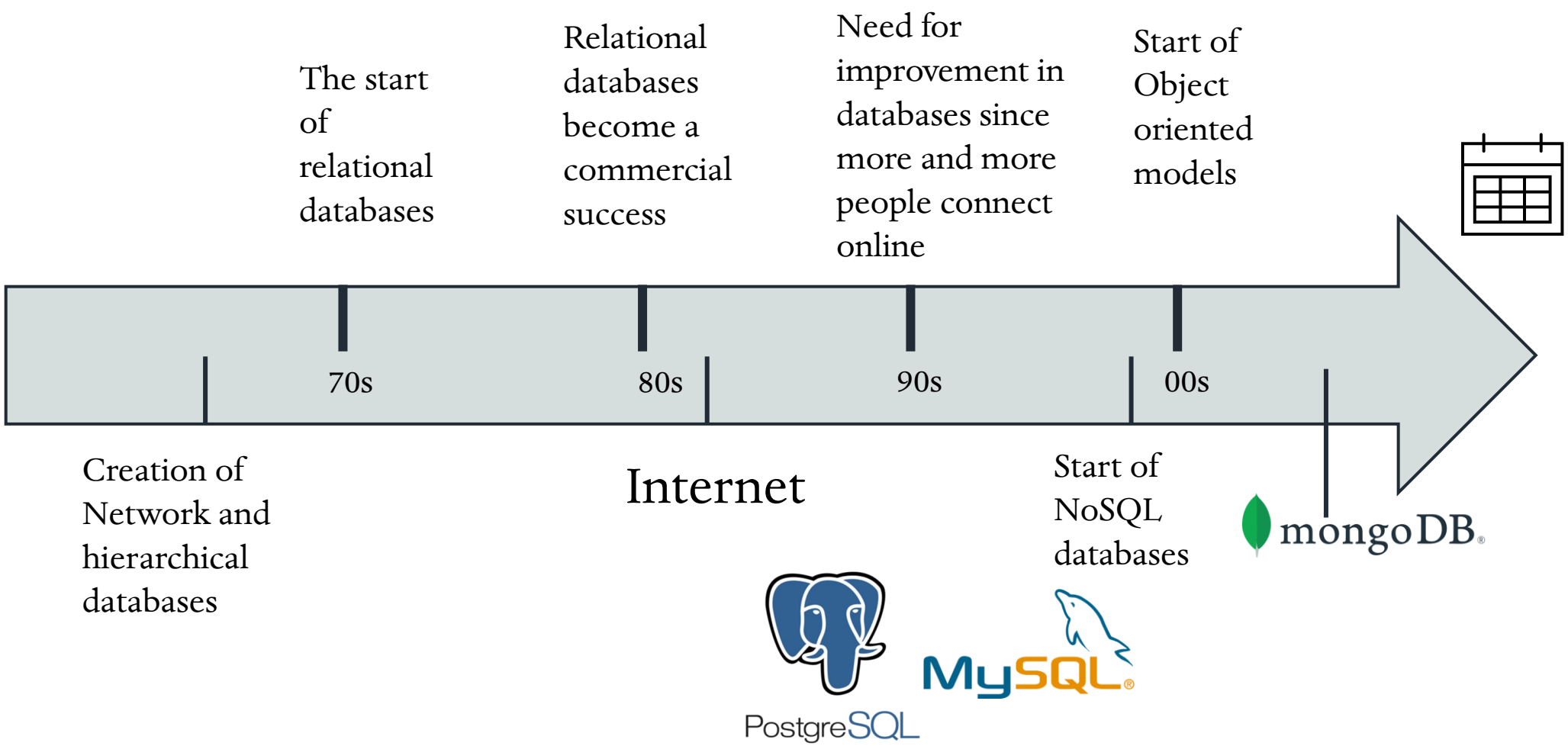
Human factors in software development

A study on database system adoption by developers

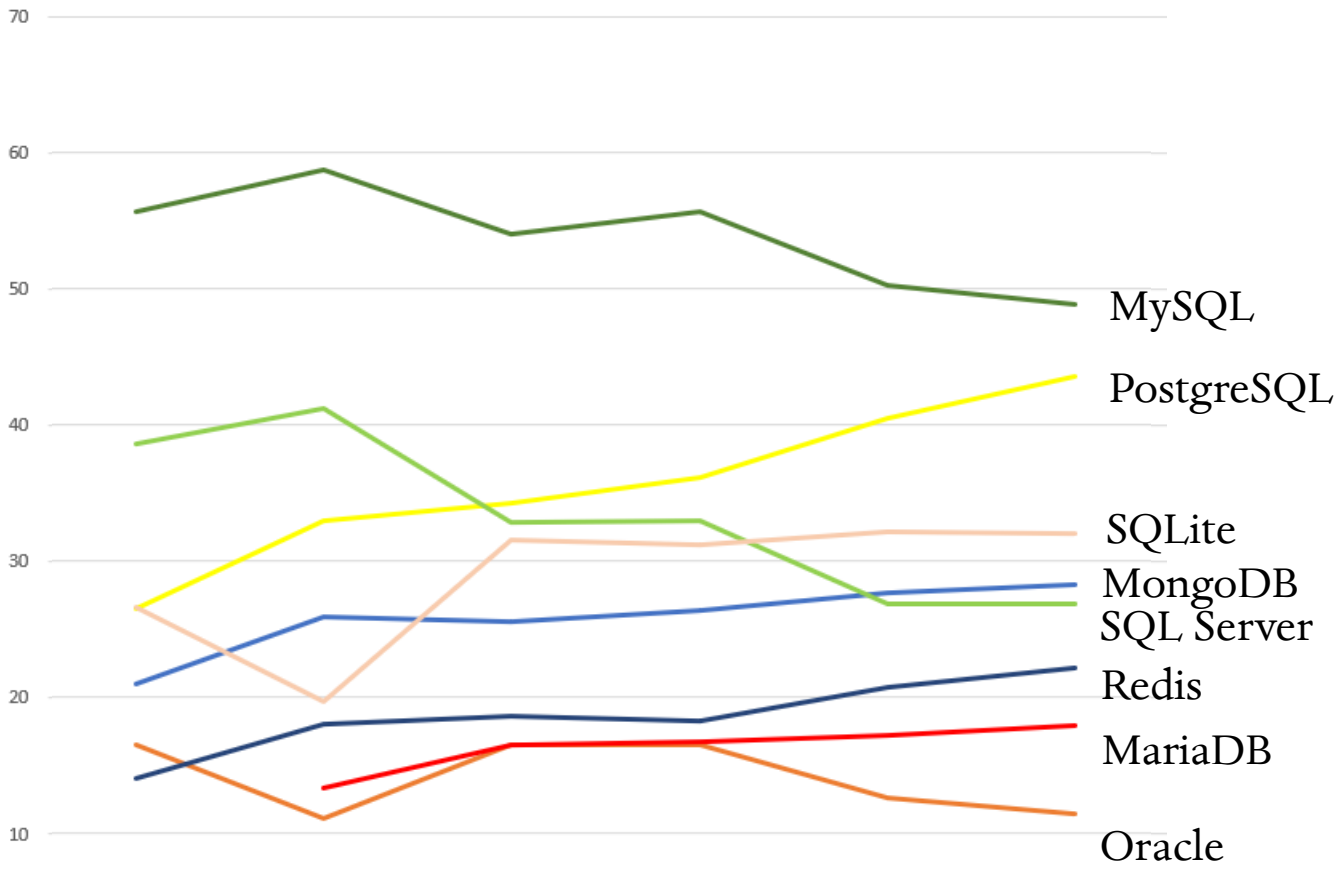
Ioanna Terzi
Human Computer Interaction Group,
University of Patras
ioanna346terzi@gmail.com

Monica Divitini
Department of Information and Computer Science,
Norwegian University of Science and Technology
divitini@ntnu.no

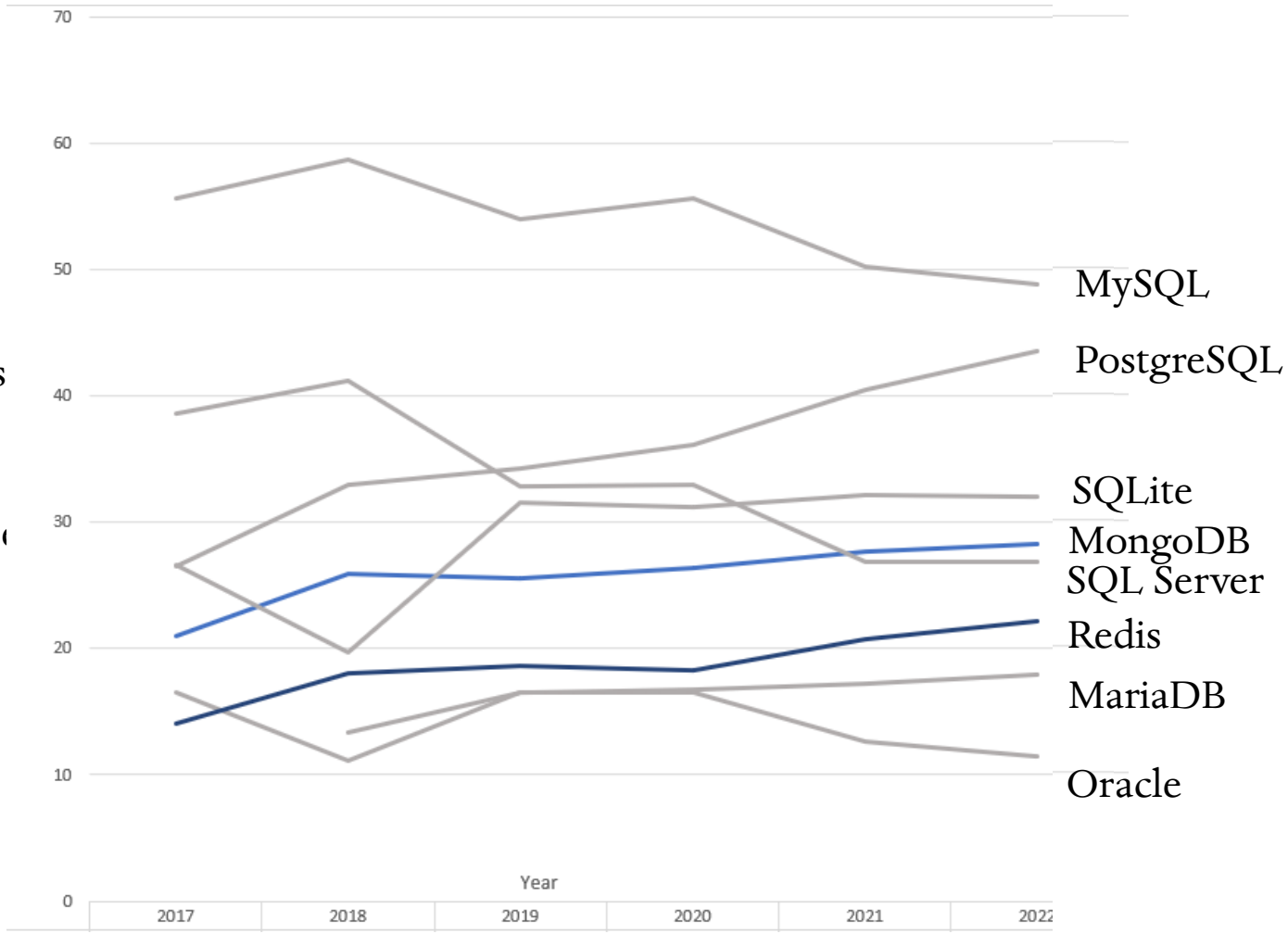
Nikolaos Avouris
Human Computer Interaction Group,
University of Patras
avouris@upatras.gr


The start of relational databases
Relational databases become a commercial success
Need for improvement in databases since more and more people connect online
Start of Object oriented models
70s
80s
90s
00s
Creation of Network and hierarchical databases
Internet
Start of NoSQL databases
mongoDB
PostgreSQL
MySQL

% of programmers that worked intensively in every database model



% of programmers that worked intensively in every database model



So…
The adoption of recent database technology is rather slow

Research questions

1. Investigate cases and criteria for using NoSQL databases in different projects.
2. Challenges companies face when switching their database model.
3. Benefits and drawbacks of using the relational and document database model.

The sample…

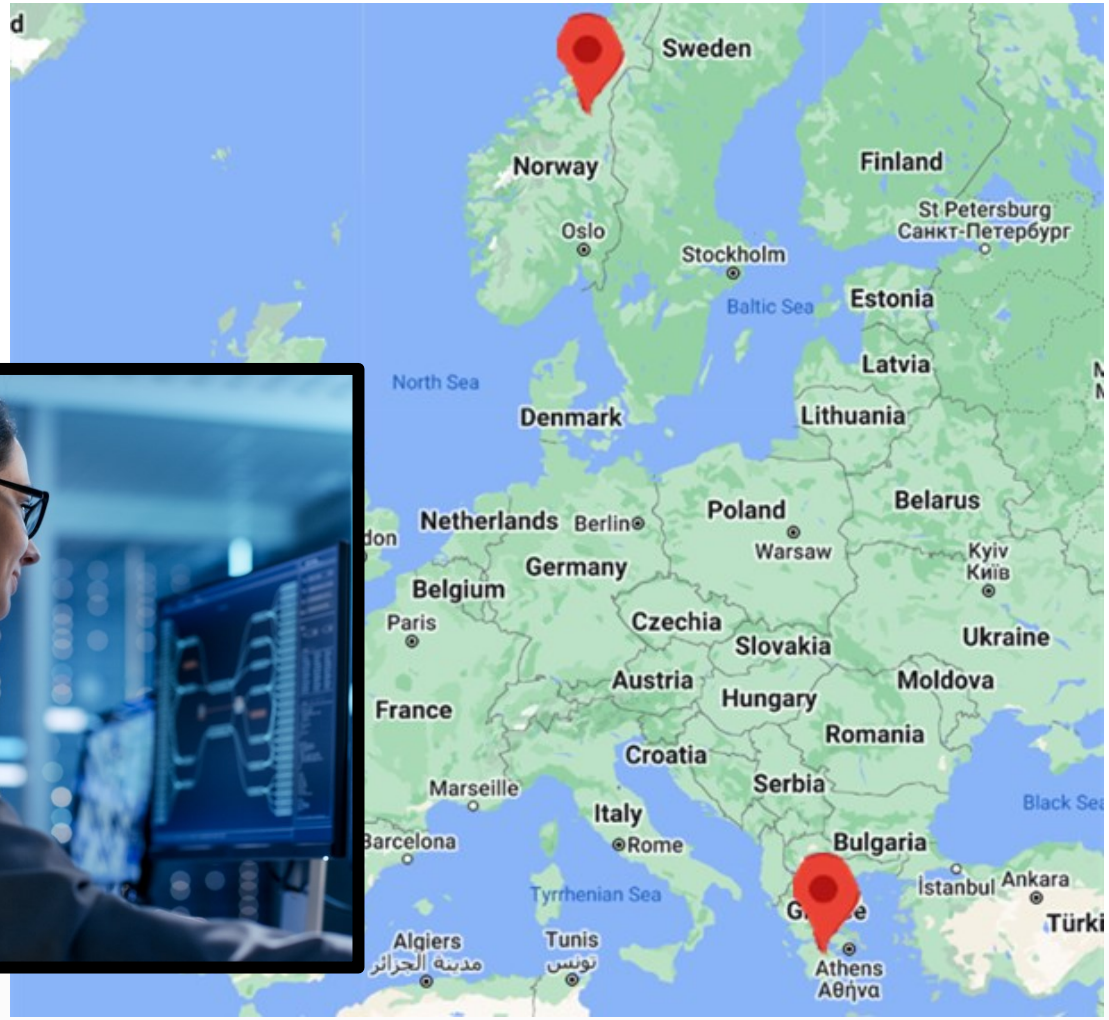
Search for typical practisioners with the following characteristics:

1) Knowledge of the field of databases
2) Work experience as IT professional

9 Participants of various levels of experience

10 Companies from Norway and Greece

Fields of Information technology and software development



Source: GORODENKOFF / SHUTTERSTOCK

Interviews were:

-Semi- structured

-Face to Face

-Audio recorded

All policies related to data protection were followed.

Source: Image by pressfoto on Freepik

Study Results

1. Programmers' selection of database in different kinds of projects

Would you consider using a NoSQL database in the following cases?

(a) Unpaid personal project

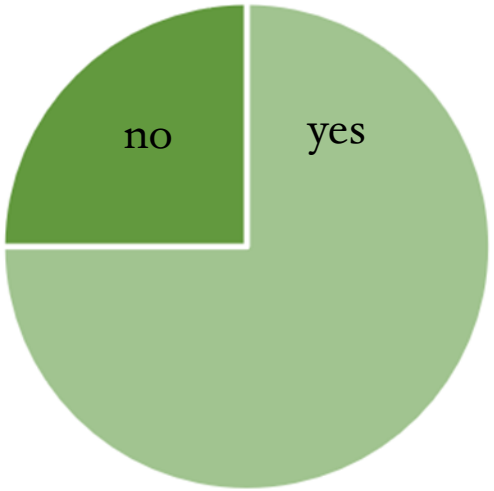
(b) Paid individual consultancy project without being employed in a company

(c) Paid project as part of the employment in a company

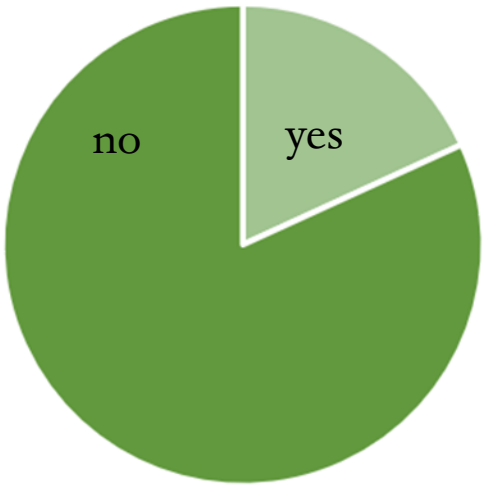
1. Programmers' selection of database in different kinds of projects

Would you consider using a NoSQL database in the following cases?

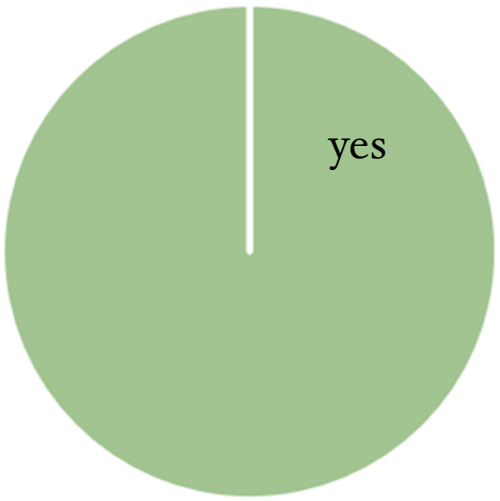
(a) Unpaid personal project



(b) Paid individual consultancy project without being employed in a company



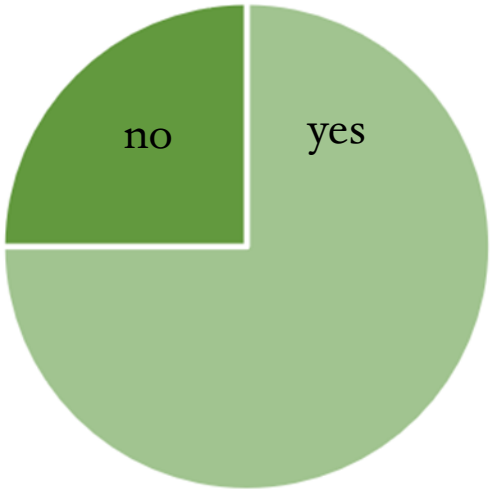
(c) Paid project as part of the employment in a company



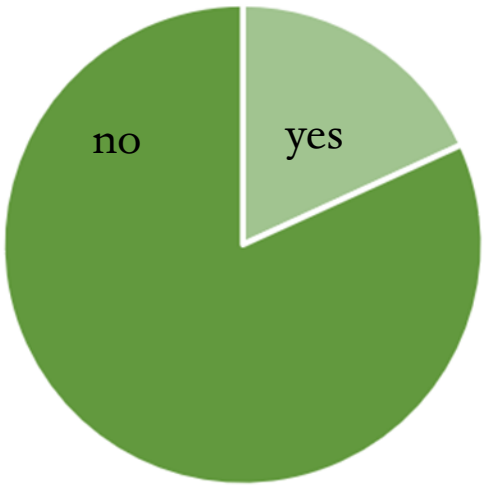
1. Programmers' selection of database in different kinds of projects

Would you consider using a NoSQL database in the following cases?

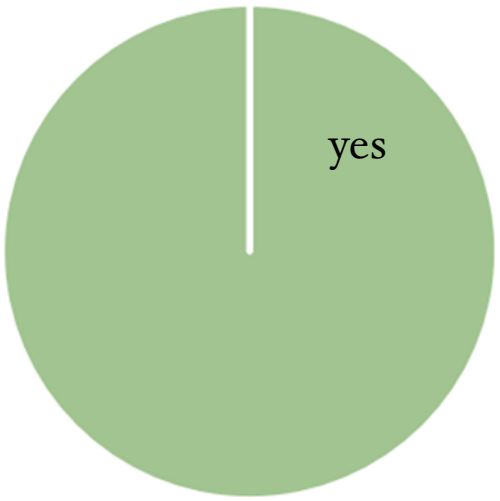
(a) Unpaid personal project



(b) Paid individual consultancy project without being employed in a company



(c) Paid project as part of the employment in a company



2. Criteria of database adoption

Which are the criteria you consider when selecting the DBMS for a professional project?

2. Criteria of database adoption

Which are the criteria you consider when selecting the DBMS for a professional project?

1. Data types

Structured vs Unstructured data

2. Popularity of the model

Important for marketing reasons (experienced particiants) and available resources

'I don't want to be a guinea pig, its important to find answers quickly'

3. Programmer's background

Previous experience with the model is important

4. DBMS functionality

There are analytical pre-installed tools in some DBMSs

5. Getting started

Minimal effort for the set up and the design of the database

6. The domain of the application

The more interrelated the data, the more the relational model fits.

'For entities that are hierarchical, one directional, MongoDB is good. For a two-dimensional connection, a relational choice would be a better choice.'

7. Budget

8. Technological context

9. Scalability

'It just scales easily to multiple nodes'

3. Company's technology adoption

Why do companies appear to be hesitant to change their data model?

3. Company's technology adoption

Why do companies appear to be hesitant to change their data model?

1

Human resources

Raises the issue of training employees or hiring new ones.

2

Data sensitivity

The less contact with the data, the more their confidentiality is ensured.

3

Transition time

For the 2 companies that attempted it, it was 6 months and 2 years respectively.

4

Operational continuity

Avoiding downtime.

5

Management approval

6

Cost

4. Database perception by programmers

In your experience, what are the main benefits and drawbacks of using the relational and the document database model?

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

Contradiction 1

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much ti
(+) Simpler model.	(-) Not user-
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	More programmable.
(+) DBMSs are more mature.	(performance.
(+) Supports understanding of other models.	web data, html.
(+) Easy to connect to programming languages.	s well in distributed
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

Contradiction 1

'It requires a lot of engineering to build a good SQL, I believe people without university knowledge find it scary'

'Simplicity was the selling point of SQL when it was launched'

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

Contradiction 2

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	
(-) Challenging at first.	itive but there
(-) ERDs are hard.	umentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

'There are 3 potential interfaces to ana application: UI, programming and SQL. MongoDB offers all 3, while relational DBMSs are more restricted and require to download a plug-in for the other interfaces

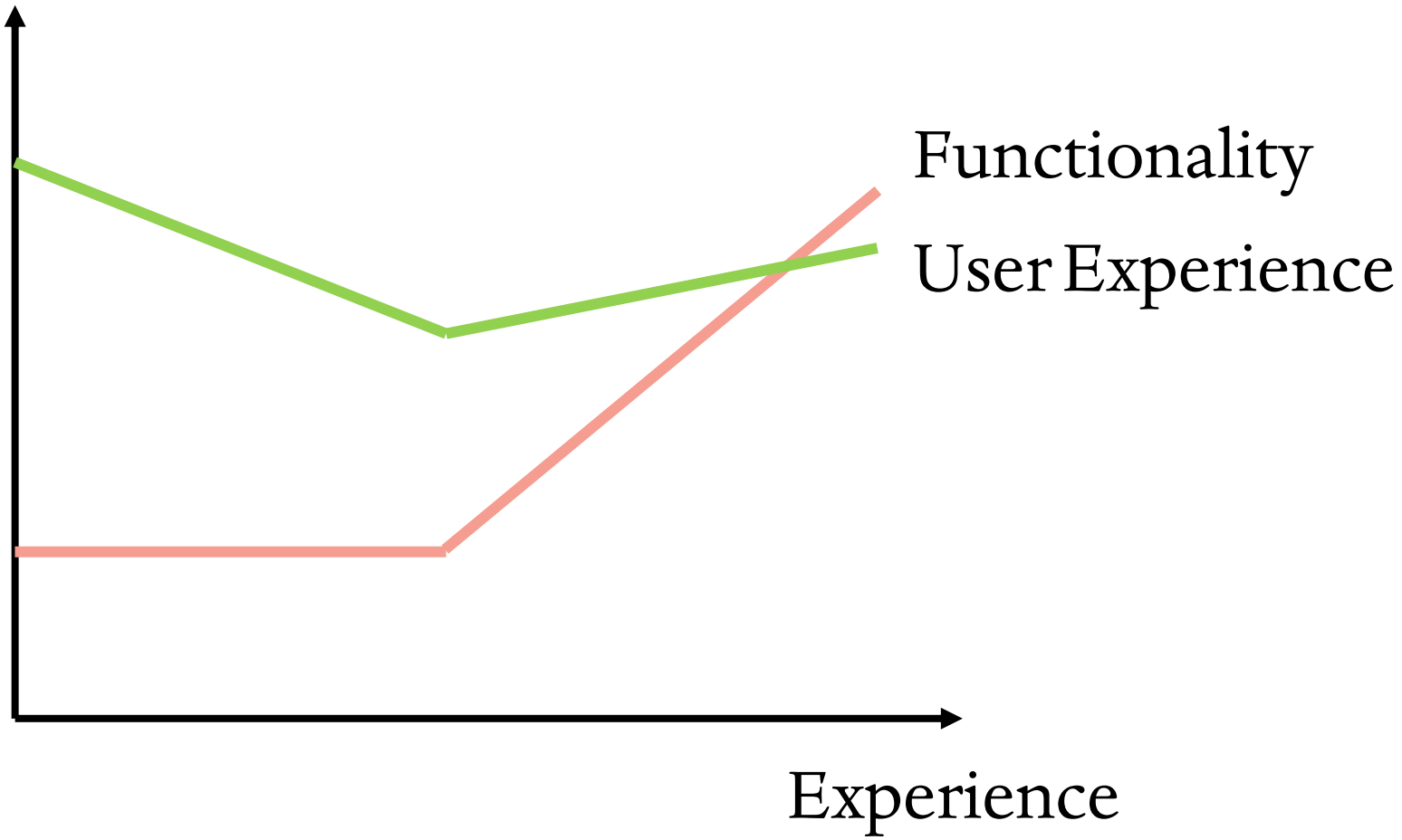
Contradiction 2

Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

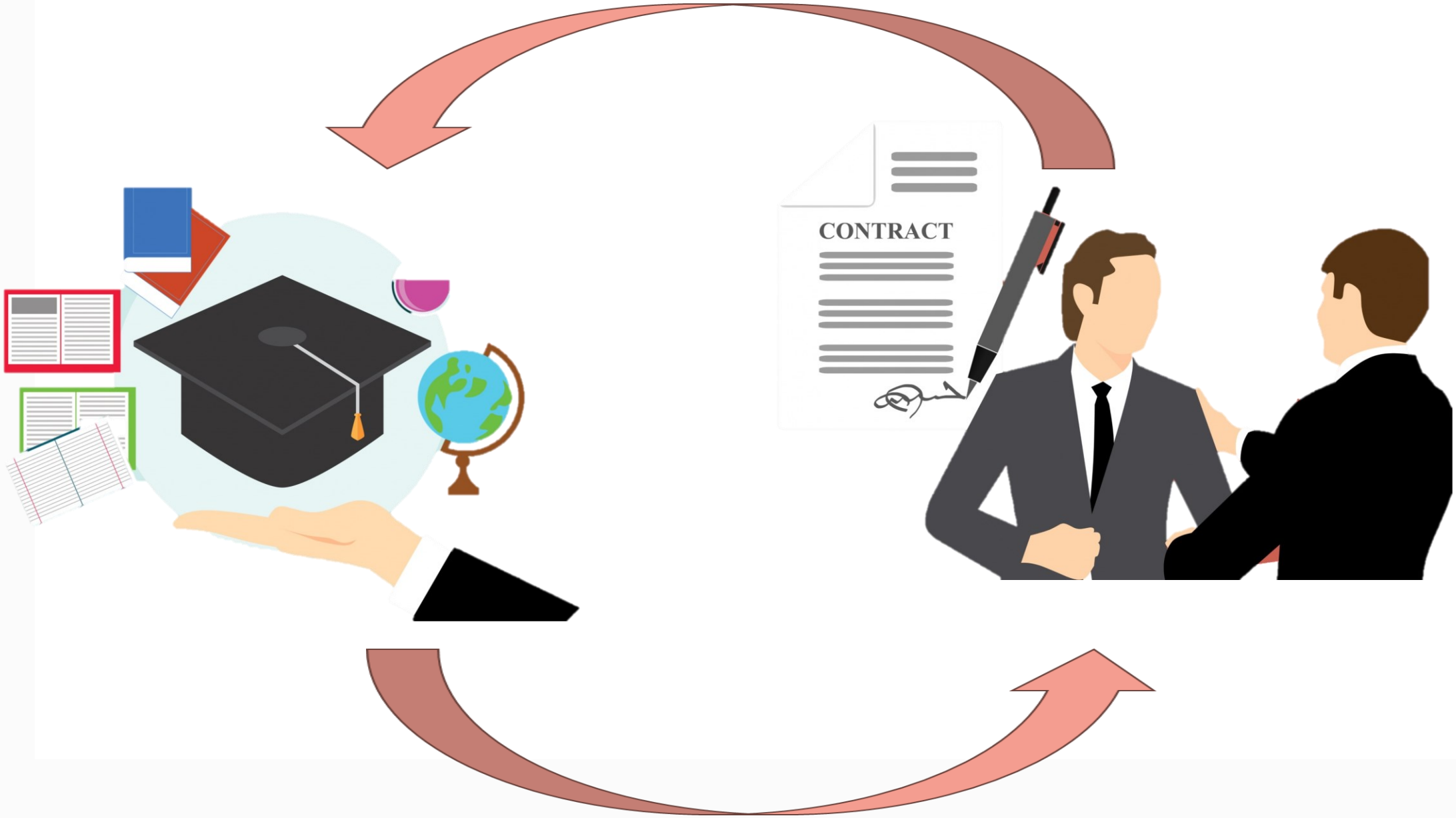
Relational model	NoSQL-Document Model (MongoDB)
(-) More complex and difficult model.	(-) Much time needed to familiarize to JSON data type, uncomfortable syntax.
(+) Simpler model.	(-) Not user-friendly environment.
(-) Challenging at first.	(+) Installation and administration are more intuitive but there is small difference.
(-) ERDs are hard.	(-) Frequent changes, that make you need the documentation.
(+) Good visualization.	(+) Ability to extract whole documents without filtering.
(-) More time consuming to set it up.	(+) More programmable.
(+) DBMSs are more mature.	(+) Better performance.
(+) Supports understanding of other models.	(+) Ideal choice for JSON data, web data, html.
(+) Easy to connect to programming languages.	(+) It's more scalable and works well in distributed environments.
(-) Not good option for hierarchies, trees, graphs.	(+) Better for connections with one-direction.
(-) For complex problems the solutions become complex in SQL because of nested queries.	(-) Lack of transactions support.
	(-) Required more resources (memory) in comparison to MySQL.

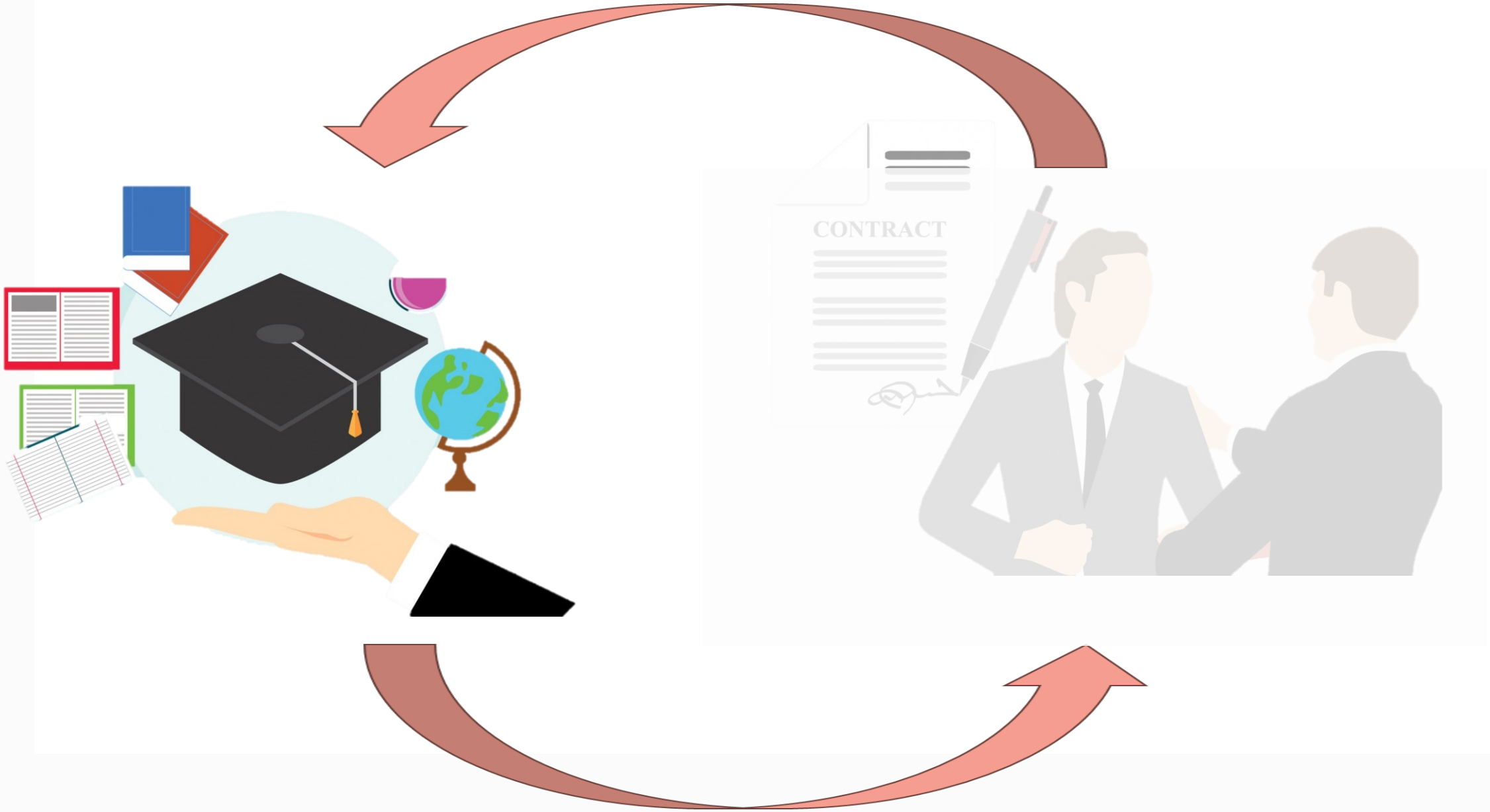
Aggreement

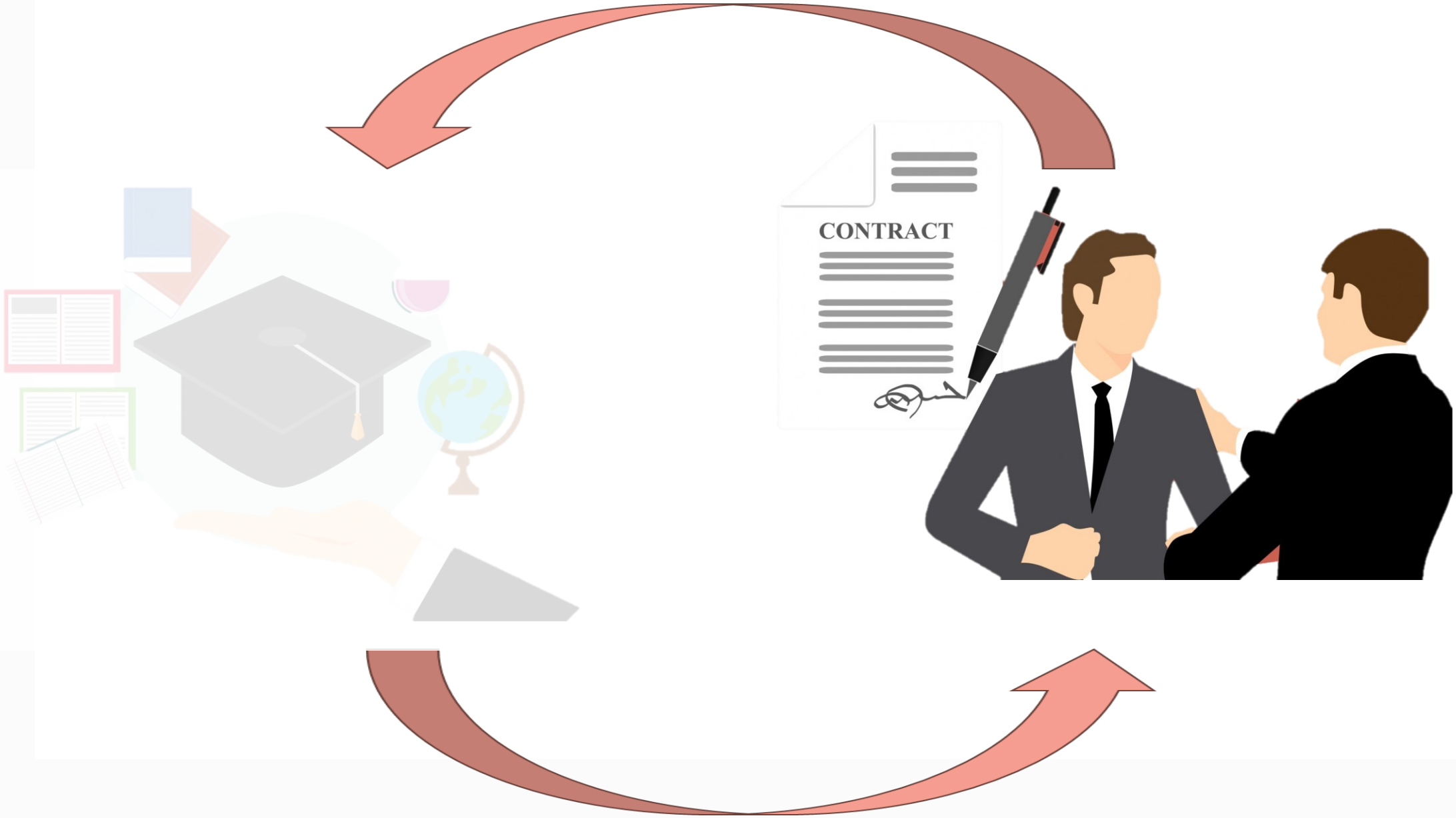
Discussion


Functionality
User Experience
Experience

So,
about the prevailance of the relational model…


CONTRACT


CONTRACT


CONTRACT

Thank you!

For more details, please read our study's paper.

Source: Image by eightonesix on Freepik

Question time!