




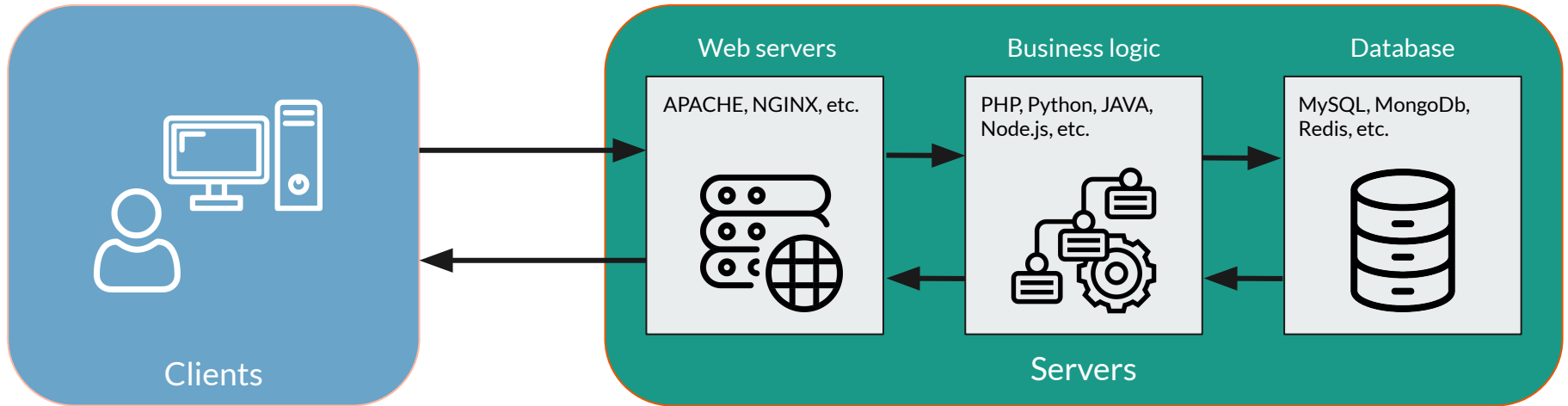
NoSQL Database

Introduction

- 
1. What is a database?
 2. SQL Database and ORM
 3. SQL vs NoSQL

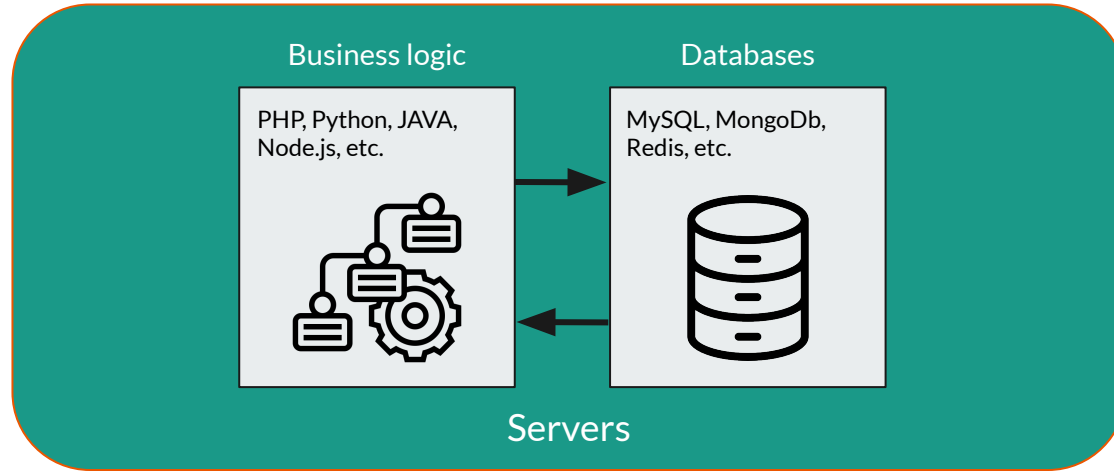
What is a database?

Architecture



What is a database?

Architecture



What is a database?

SQL Database and ORM

SQL?



SQL is not a database, is a language to write database queries

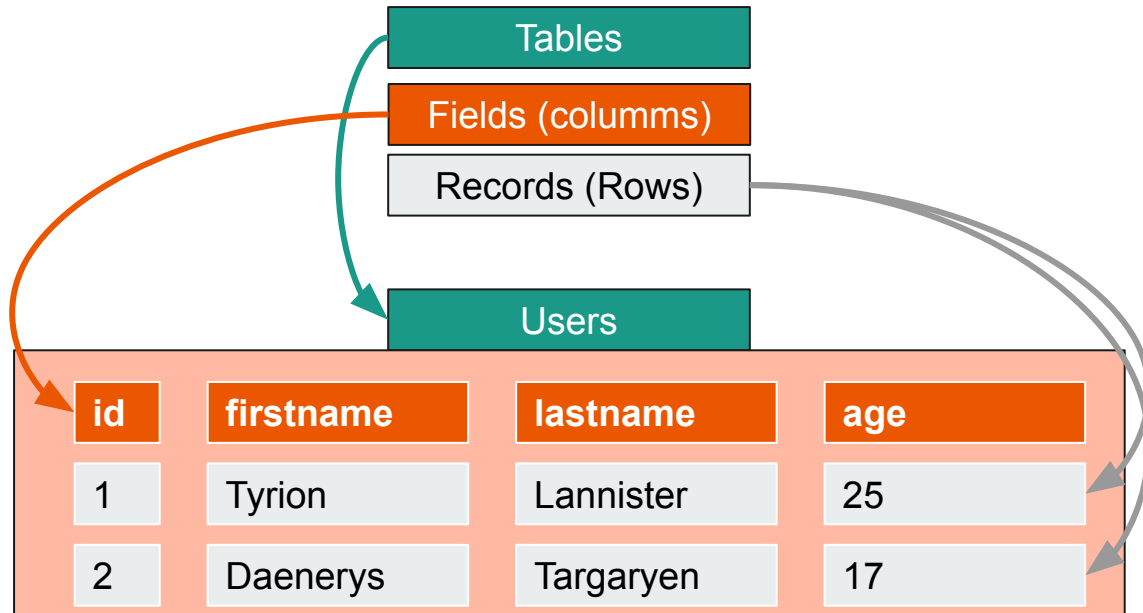
Structured Query Language

SELECT id, firstname, age **FROM** users

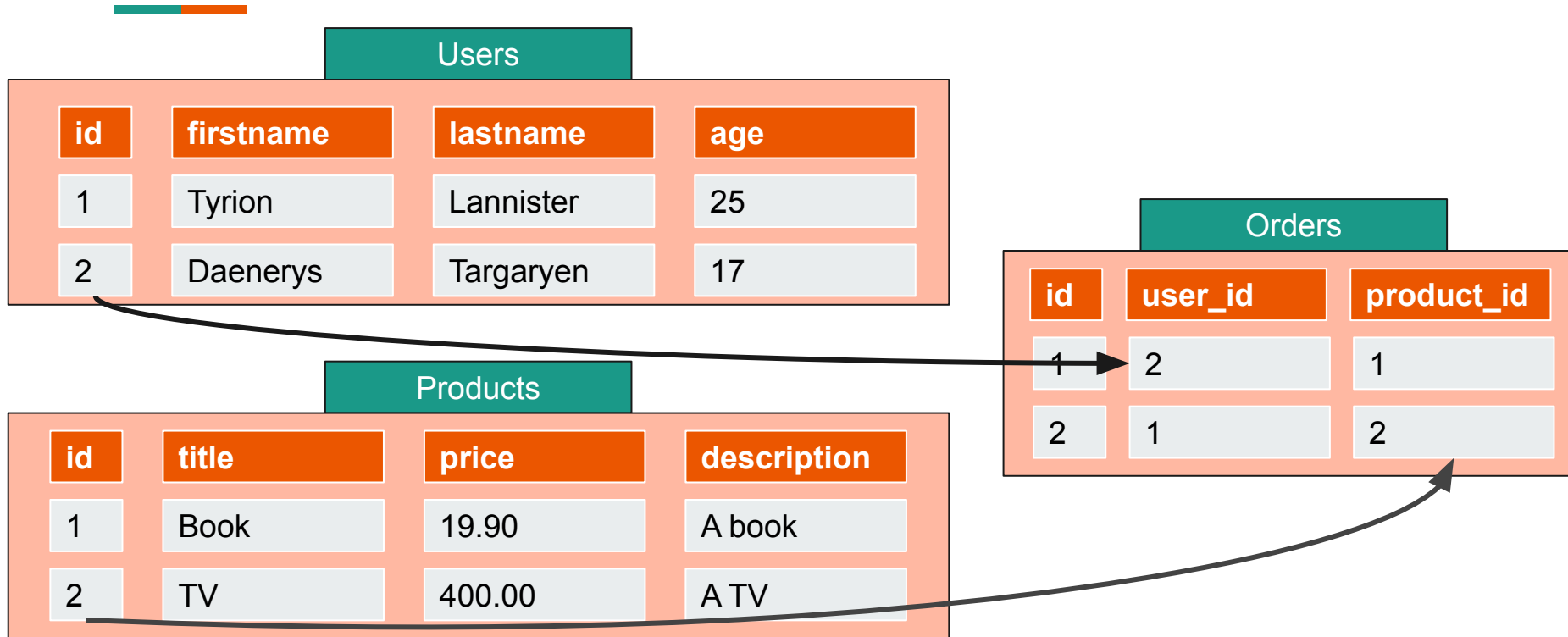
keywords: *SELECT, INSERT, FROM, etc.*

SQL database: relational database

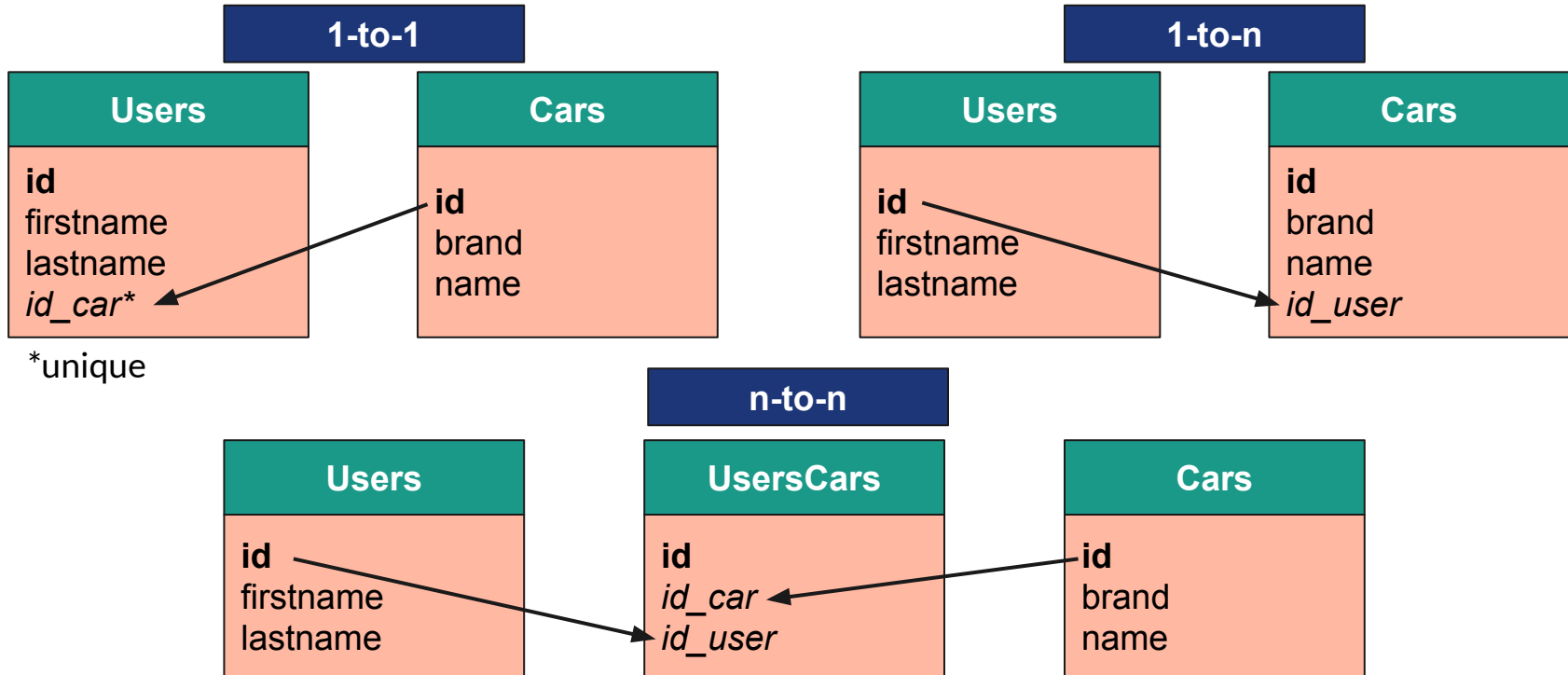
Clear schemas with fixed Fields (columns)



Relations



Types of Relations



SQL - characteristics



Strict schemas

and

Relations

+

Index

Project: MySql, Oracle, Sqlite, Postgres and MS-SQL.

What is an ORM?



- **Object-Relational Mapping or ORM**

Makes the database relationship to Object Oriented

- Writing classes leads to the creation of tables
 - No SQL query - use of methods
 - It is the ORM that makes the requests.
-
- **Warning:** not always optimized

ORM in JAVA

```
public class Employee {  
    private int id;  
    private String first_name;  
    private String last_name;  
    private int salary;  
  
    public Employee() {}  
    public Employee(String fname, String lname, int salary) {  
        this.first_name = fname;  
        this.last_name = lname;  
        this.salary = salary;  
    }  
  
    public int getId() {  
        return id;  
    }  
  
    public String getFirstName() {  
        return first_name;  
    }  
  
    public String getLastName() {  
        return last_name;  
    }  
  
    public int getSalary() {  
        return salary;  
    }  
}
```



```
create table EMPLOYEE (  
    id INT NOT NULL auto_increment,  
    first_name VARCHAR(20) default NULL,  
    last_name VARCHAR(20) default NULL,  
    salary INT default NULL,  
    PRIMARY KEY (id)  
);
```

SQL vs NoSQL

SQL vs NoSQL



SQL

Strict schemas

Relations

Tables

Limitation read/write

Vertical scaling

NoSQL

Performance for read/write

Horizontal scaling

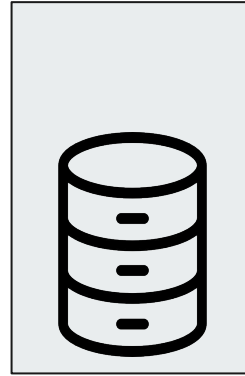
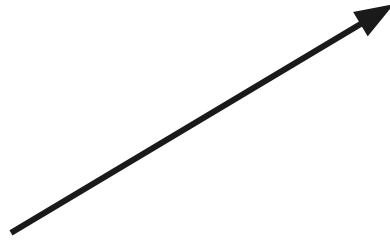
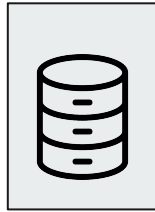
Many types

Scaling

Scaling



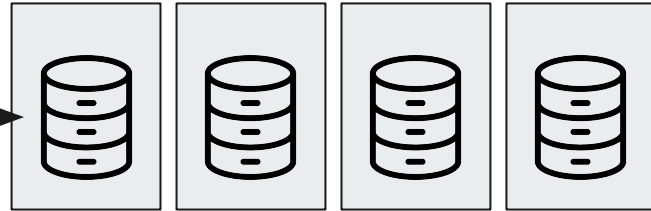
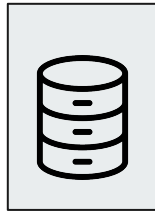
Vertical Scaling
SQL database



more CPU
more RAM
more HDD

But one machine -> limits

Horizontal Scaling
NoSQL database

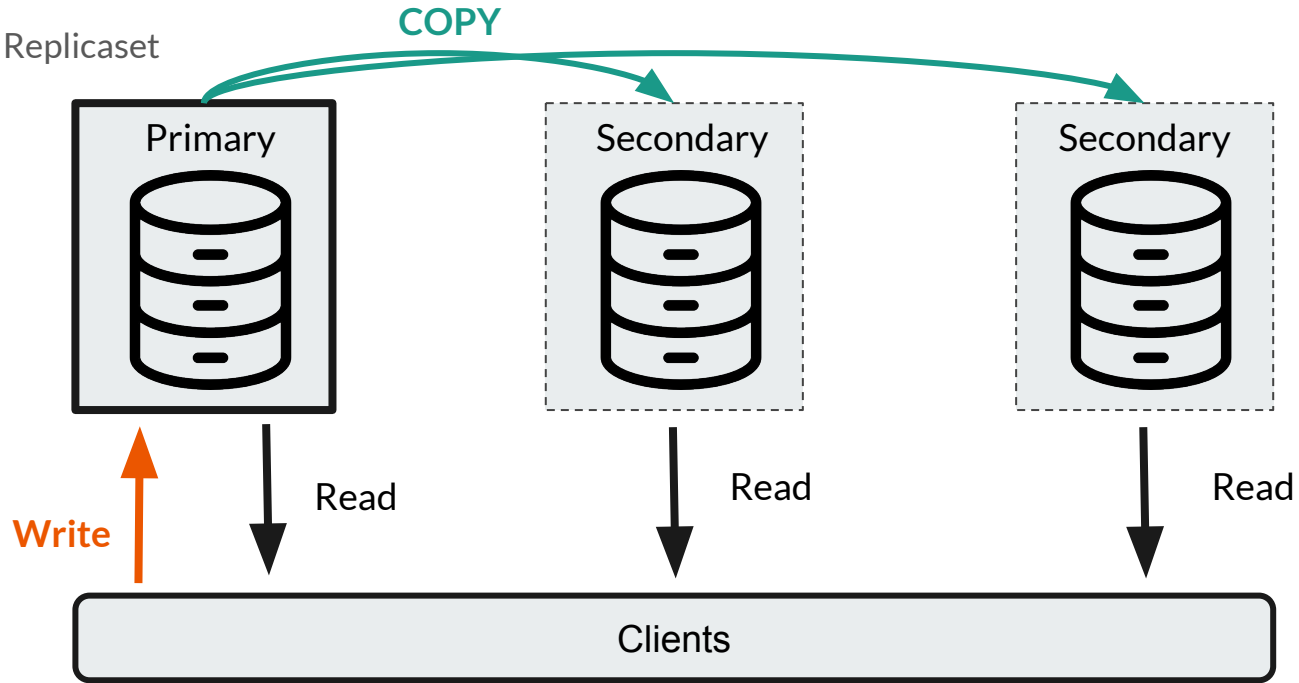


more machines: no limits

Scaling

Advantages of Horizontal Scaling

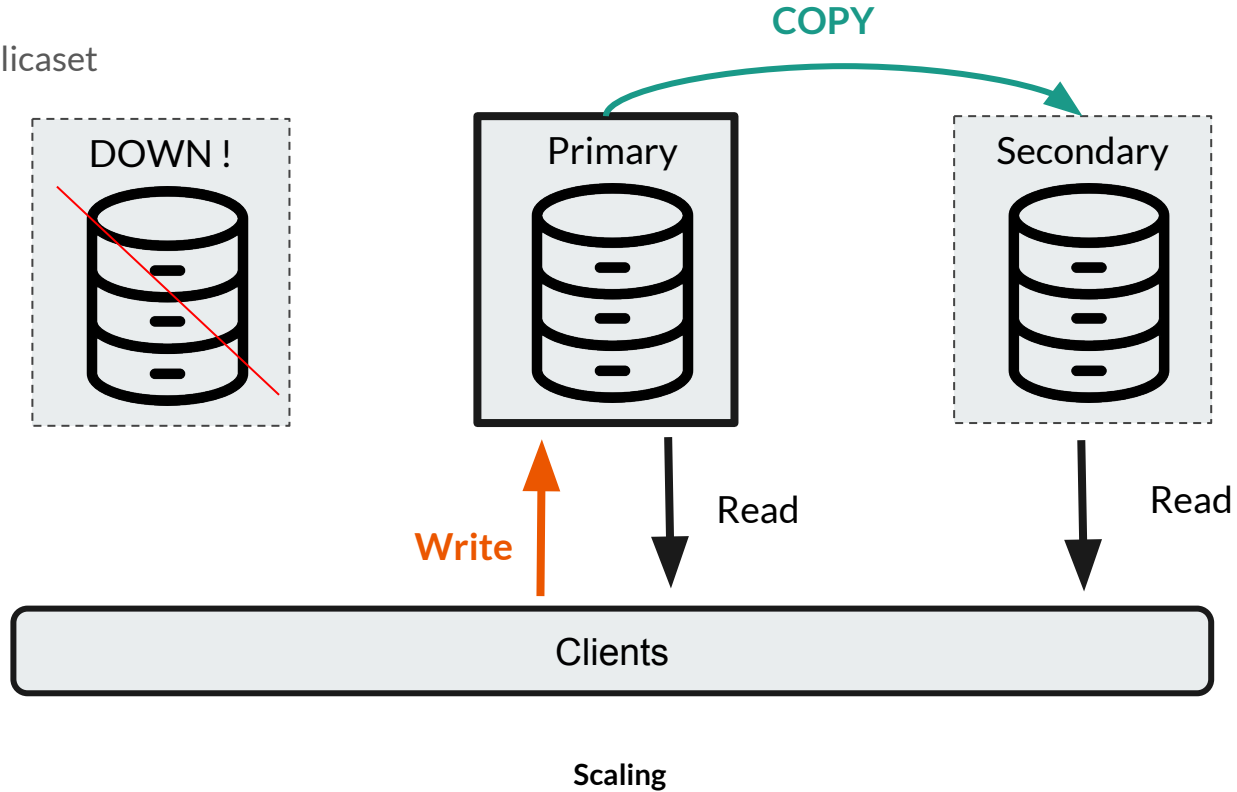
1. Replicaset



Scaling

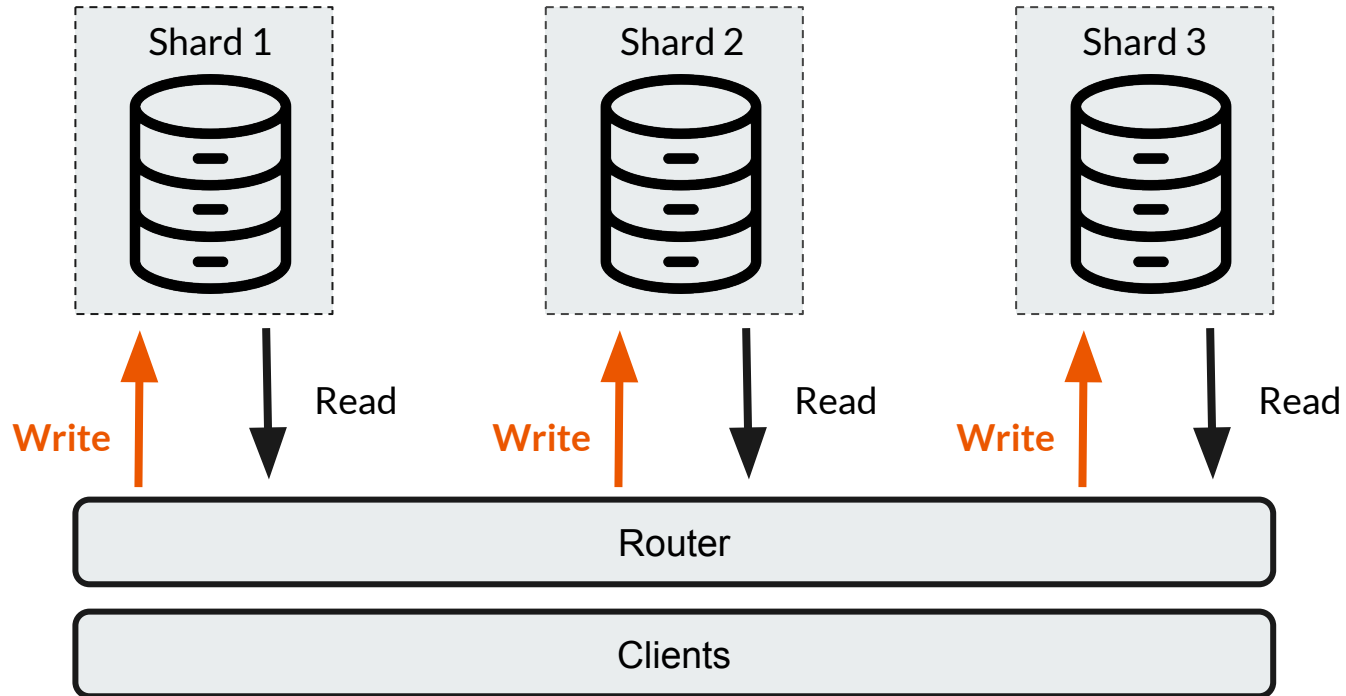
Advantages of Horizontal Scaling

1. Replicaset



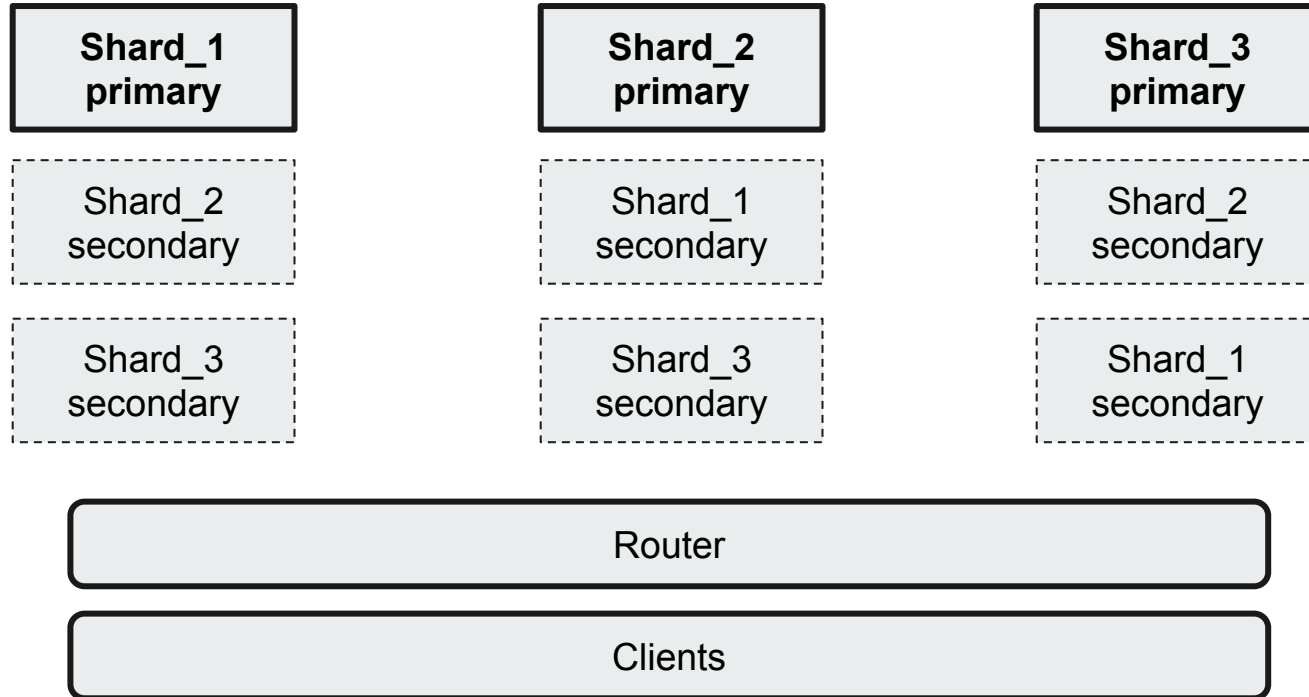
Advantages of Horizontal Scaling

2. Sharded: by region, id, other



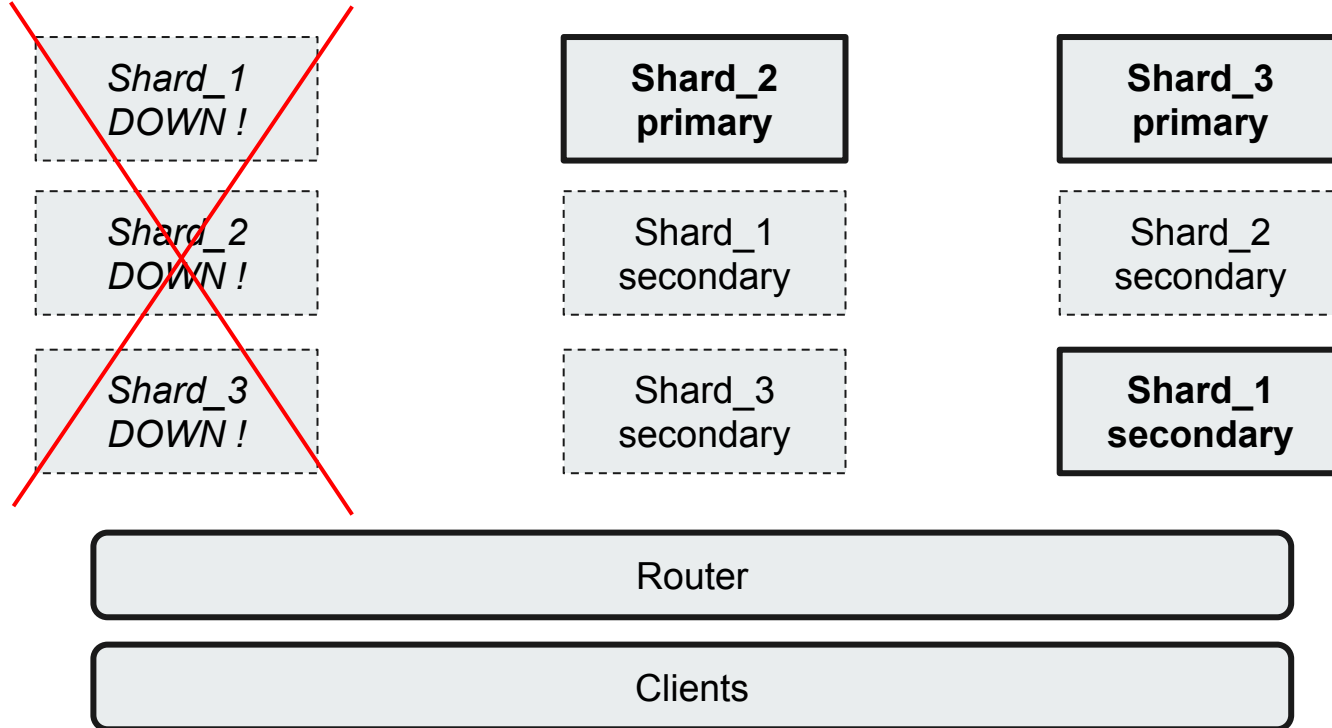
Advantages of Horizontal Scaling

IRL



Advantages of Horizontal Scaling

IRL



NoSQL

Key-value, columns, document,
graph

NoSQL

1. keys-values

Key-Value database



key	value
user:1	"Tyrion"
user:2	"Daenerys"
nbr_users	50
...	...
...	...

Key-Value database



- Use RAM for storage
- not always persistent DATA
- Use for
 - catching DATA
 - temporary code
 - USER session
 -

Key-Value database



exemple: REDIS

- SIMPLE
- Structured DATA: List, Set, Map, SortedSet
- Can have persistent DATA (store every x write or log)
- MASTER - SLAVE (real time copy)

Key-Value database

REDIS: DATA type

	Keys	Values
String	user:123	{ "firstname": "Tyrion" }
List	Page:view	[nic, tom, nic, bob, anna, nic]
Hash	user:Romain	firstname => Romain lastname => Tribout
Set	student:ISEN	{nic, tom, anna}
SortedSet	votes:NoSQL	{ bob => 5 tom => 8 anna => 6 }

Key-Value database

Key-Value database

REDIS: basic instructions

Instructions	Description
SET <key> <value>	Create key-value
GET <key>	Read key-value
INCR <key> or DECR <key>	Increment or decrement value
TTL <key>	Get time to live
EXPIRE <key> <ttl>	Set time to live

Key-Value database

REDIS: list

Instruction	Description
R PUSH <key> <value> or L PUSH <key> <value>	Push data right or left in list
L RANGE <key> <from_index> <to_index>	get list data
L LEN <key>	Size of list
L POP <key> or R POP <key>	Remove right or left dat in list

Key-Value database

REDIS: SET (like LIST with unique)

Instruction	Description
SADD <key>	Add data to the SET
SMEMBERS <key>	Get all data
SREM <key> <value>	Remove data to the SET
SISMEMBERS <key> <value>	Data is in SET
SUNION <key1> <key2>	Union of two SET

NoSQL

2. Columns databse

Columns database



- Similar to SQL database: structured data
- Distributed (plusieurs noeuds) - big cluster = security
- Query language look like SQL
- Scalability

Columns Database



Column

name
value
<i>timestamp</i>

Columns database

Columns Database

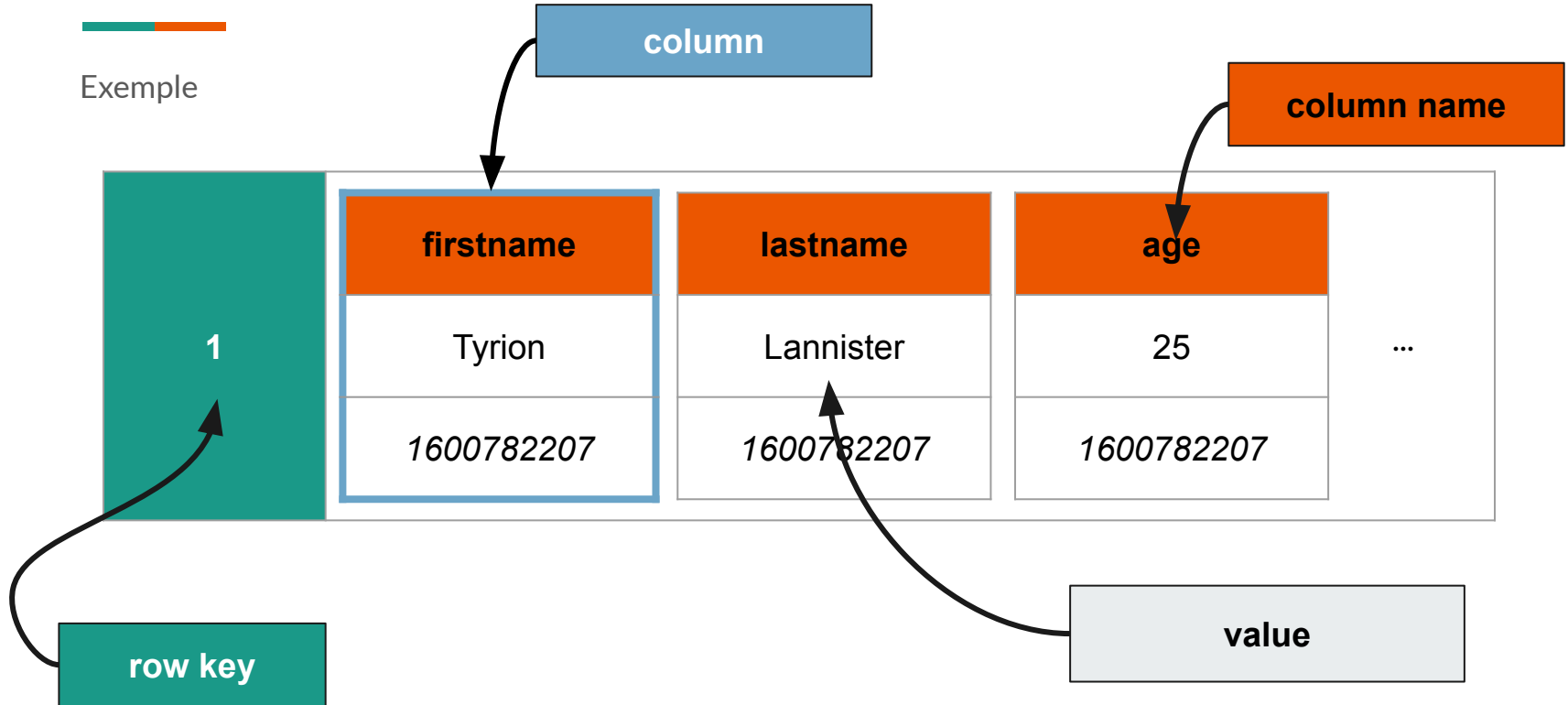
Row

Key	name 1	name 2	...	name n
	value 1	value 2		value n
	<i>timestamp</i>	<i>timestamp</i>		<i>timestamp</i>

Columns database

Columns Database

Exemple



Columns database

Columns Database

liberty but comparator/validator

1	firstname	lastname	age
	Tyrion	Lannister	25

2	firstname	lastname	email	twitter
	Daenerys	Targaryen	daenerys@gmail.com	@daenerys

Columns database

Columns Database

Column Family

`user['1']['age'] = 25`

user	1	firstname	lastname	age	
		Tyrion	Lannister	25	
	2	firstname	lastname	email	twitter
		Daenerys	Targaryen	daenerys@gmail.com	@daenerys

Columns database

Columns Database

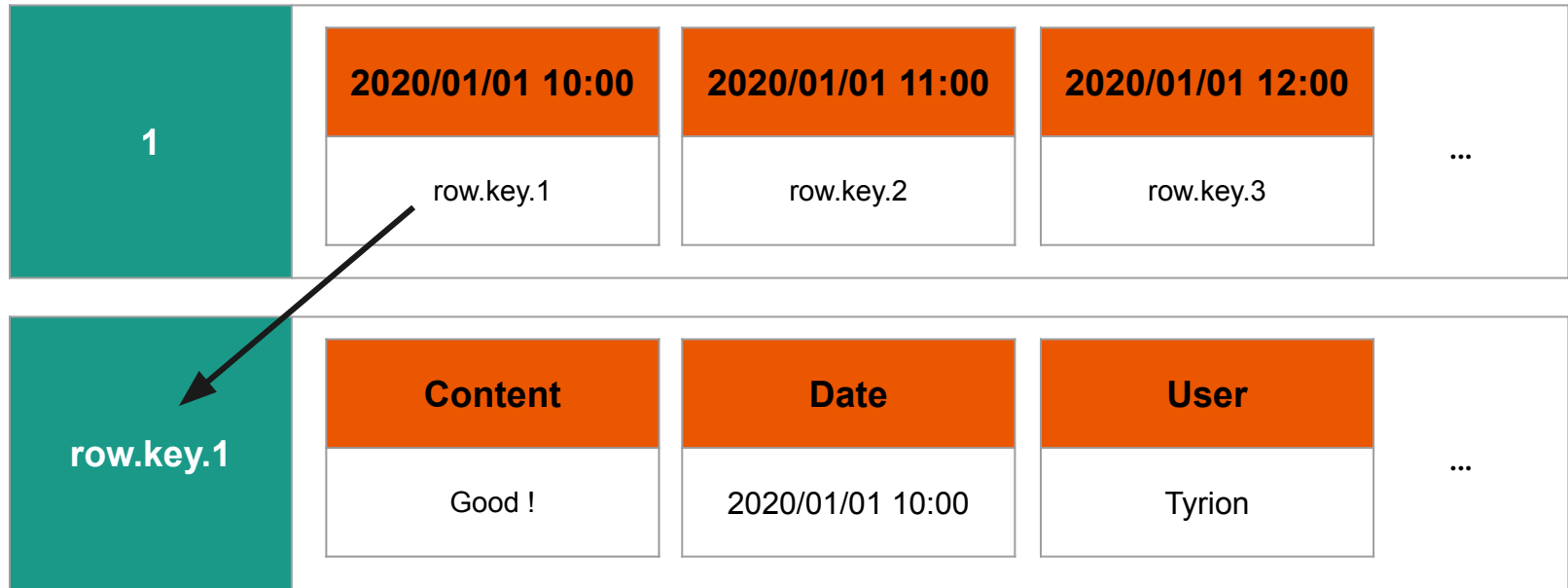
Pattern: column name can be a value

XK_4501_201 2_01_06	6:55	7:00	7:20	...
	Décollage face à la mer	survol Capo di Feno	survol Cannes	

Columns database

Columns Database

Pattern: manual relation



Columns database

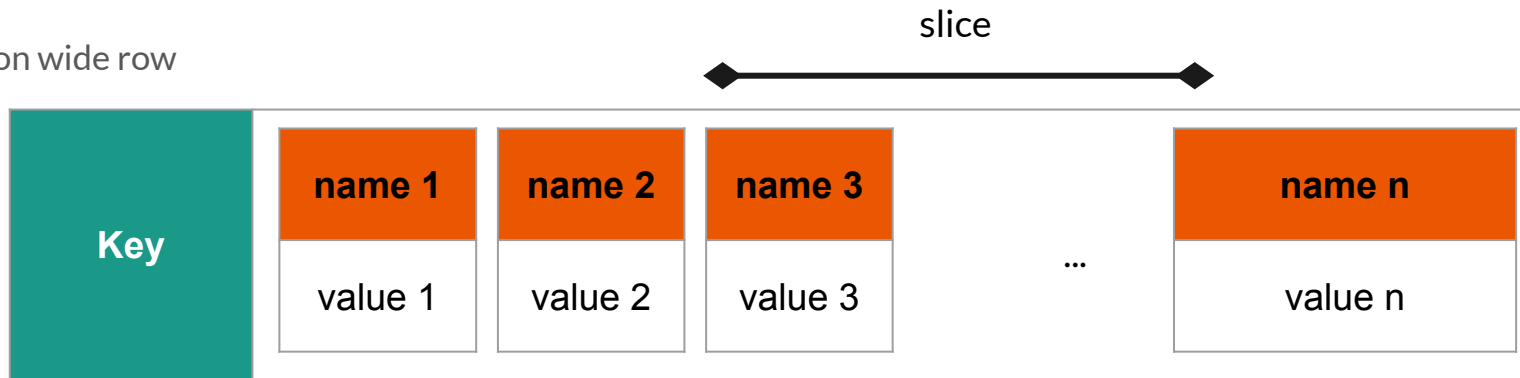
Columns Database

Queries

on skinny rows

GET: get user['1']['firstname']; or select user where country = 'fr';

on wide row



Columns database

Columns database



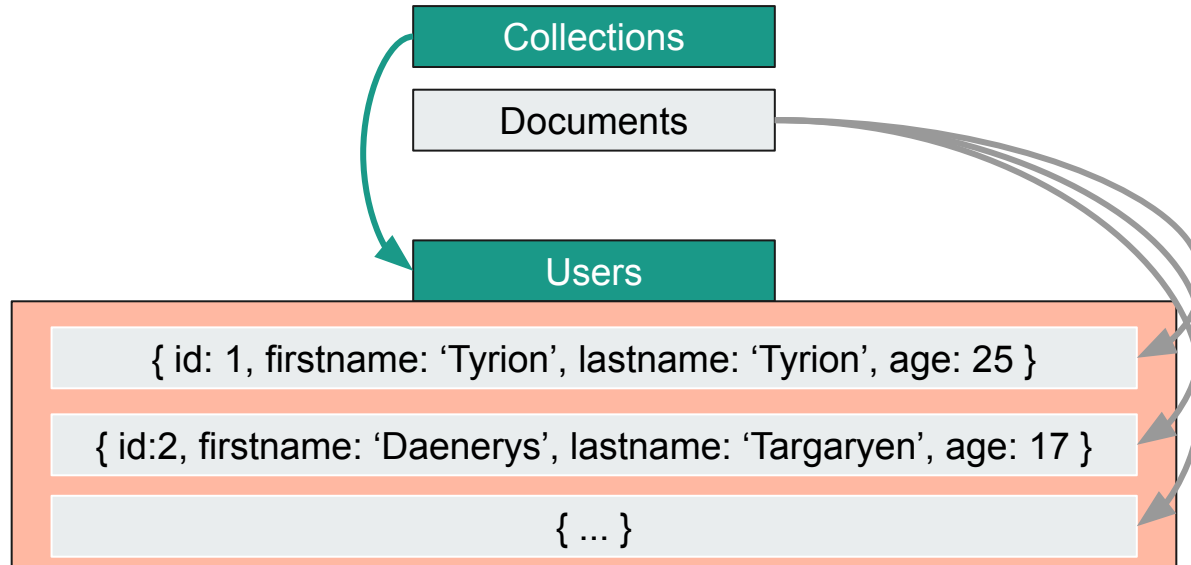
Cassandra: created by Facebook

- apple 70k nodes
- netflix 2.5k nodes

NoSQL

3. Document-oriented database

How it works



Document-oriented database

Data structure: No Schema !



Users				
1	Tyrion		25	
2	Daenerys	Targaryen	17	
3	Romain	Tribout		romain.tribout@gmail.com

Document-oriented database

No few Relations

Relations must be done manually (none-native)

Users

```
{ id: 1, firstname: 'Tyrion', lastname: 'Lannister', age: 25 }
```

```
{ id:2, firstname: 'Daenerys', lastname: 'Targaryen', age: 17 }
```

Products

```
{ id: 1, name: 'A book', price: 19.90, description: 'A book' }
```

```
{ id: 2, name: 'A TV', price: 400.00, description: 'A TV' }
```

Orders

```
{ id: 1, user_id: 1, product_id: 2 }
```

```
{ id: 2, user_id: 2, product_id: 1 }
```

Document-oriented database

No few Relations



Relations must be done manually (none-native)

Orders

```
{ id: 1, user: { id: 1, firstname: 'Tyrion', lastname: 'Lannister' }, product: { id: 2, name: 'A TV', price: 400.00, description: 'A TV' } }
```

```
{ id: 1, user: { id: 2, firstname: 'Daenerys', lastname: 'Targaryen' }, product: { id: 1, name: 'A book', price: 19.90, description: 'A book' } }
```

Document-oriented database

Characteristics



Flexible !

Performance for big queries

Project: MongoDB, CouchDB, DocumentDB

Document-oriented database

NoSQL

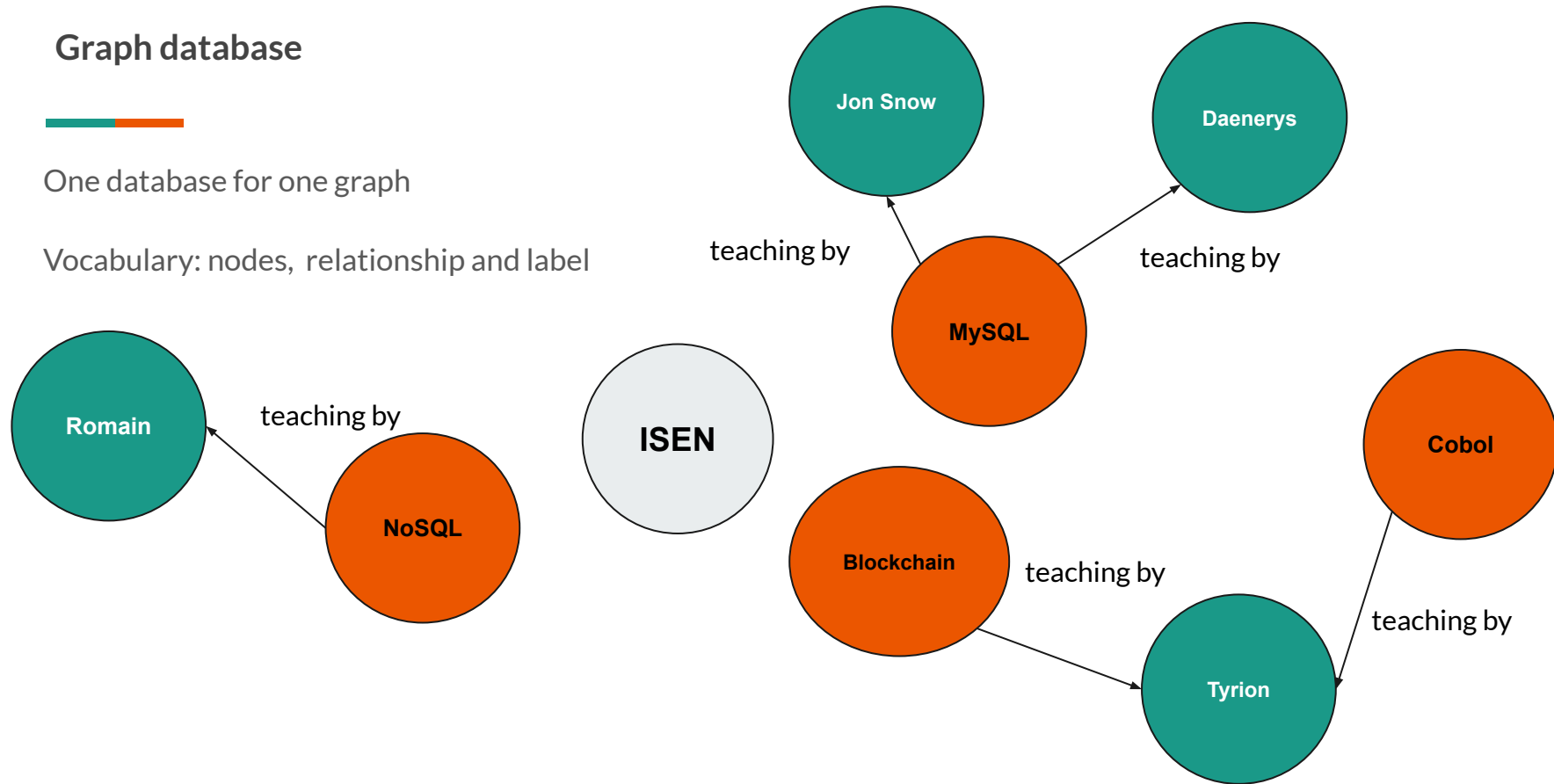
4. Graph database

Graph database



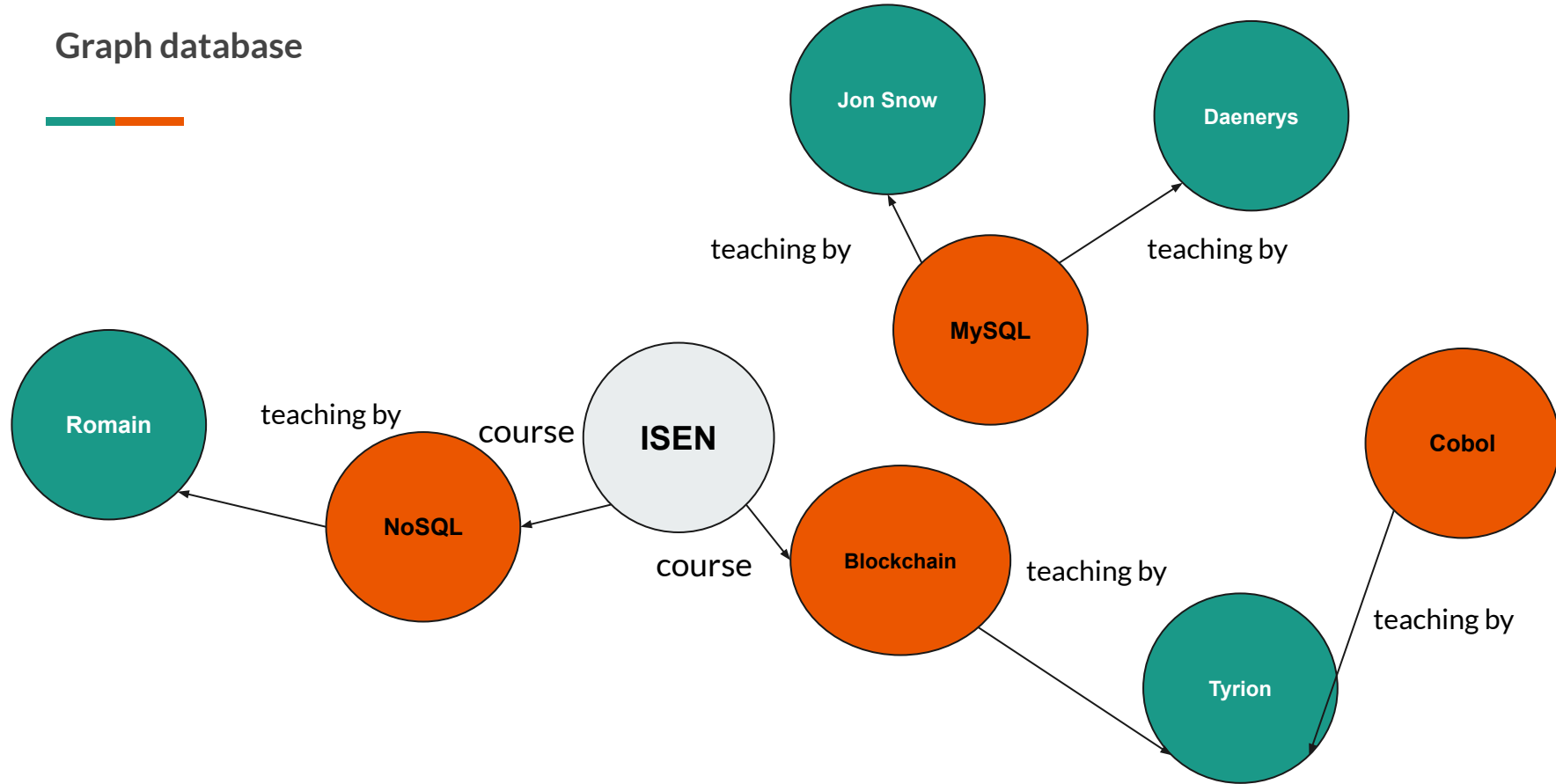
One database for one graph

Vocabulary: nodes, relationship and label



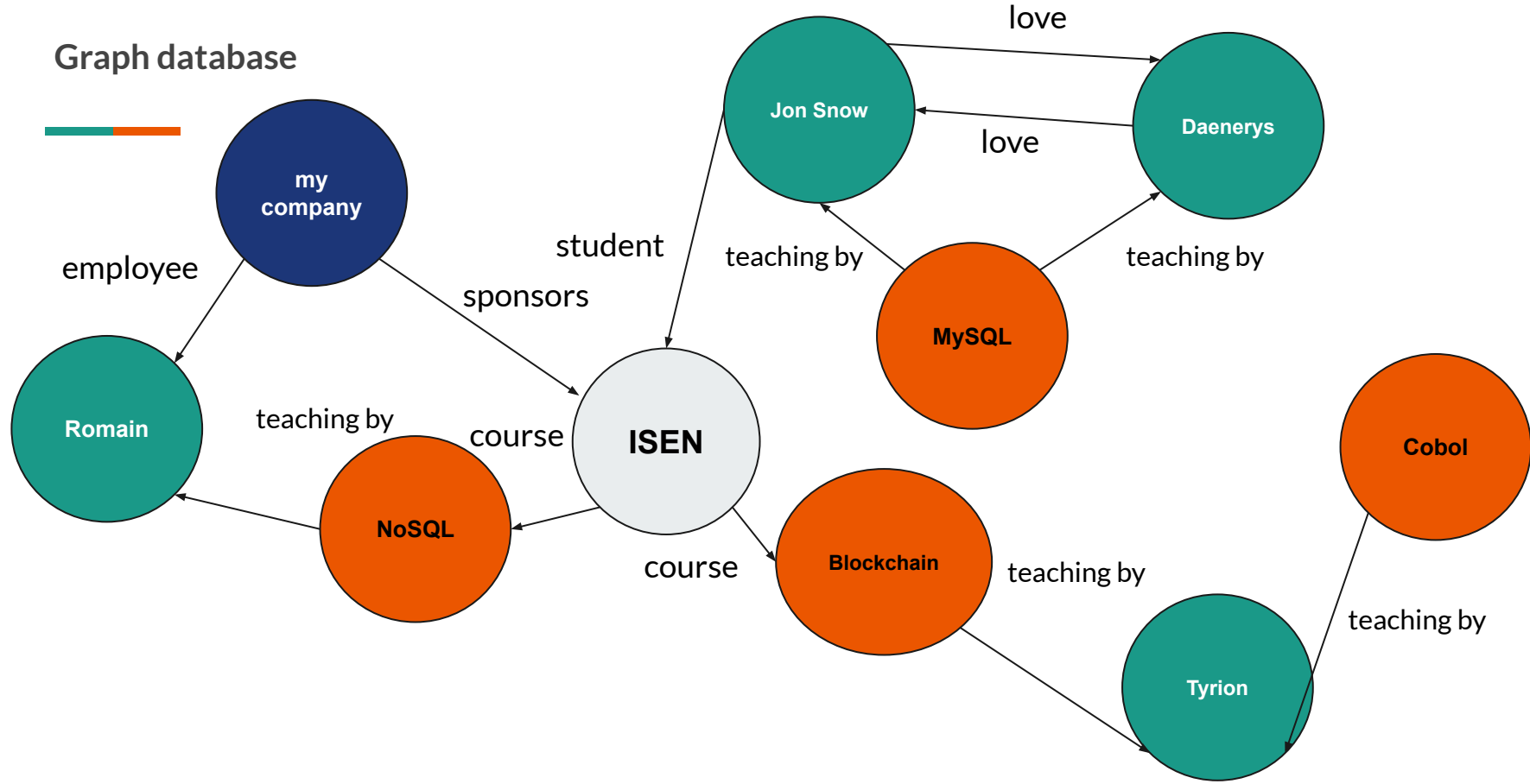
Graph database

Graph database



Graph database

Graph database



Graph database

Graph database



Request with pattern

`(LEO:PERSON)-[rel:LOVES]->(LEA:PERSON)`

`MATCH (foo:PERSON)-[rel:LOVES]->(bar:PERSON)`

`WHERE rel.duration > 5`

`RETURN foo.name bar.name rel.duration`

Graph database



project: **Neo4j**, OrientDB

Use cases

- sfr -> network graph
- meetic -> recommendations
- walmart -> recommendations
- ebay -> delivery

NoSQL and SQL

5. TIMESERIES

Timeseries



Down-sampling

Statistics

Very simple: time, value, and tags

Timeseries



project: **influxDB**, MongoDB

graph database

SQL vs NoSQL



SQL

Strict schemas

Relations

Tables

Limitation read/write

Vertical scaling

NoSQL

Performance for read/write

Horizontal scaling

Many types