# Big Data Management and Analytics
# Assignment 2

# Recap: CAP theorem

- Consistency: All clients have always the same view of the data
- Availability: Each client can always read and write
- Partition Tolerance: The system works well, despite physical network partitions

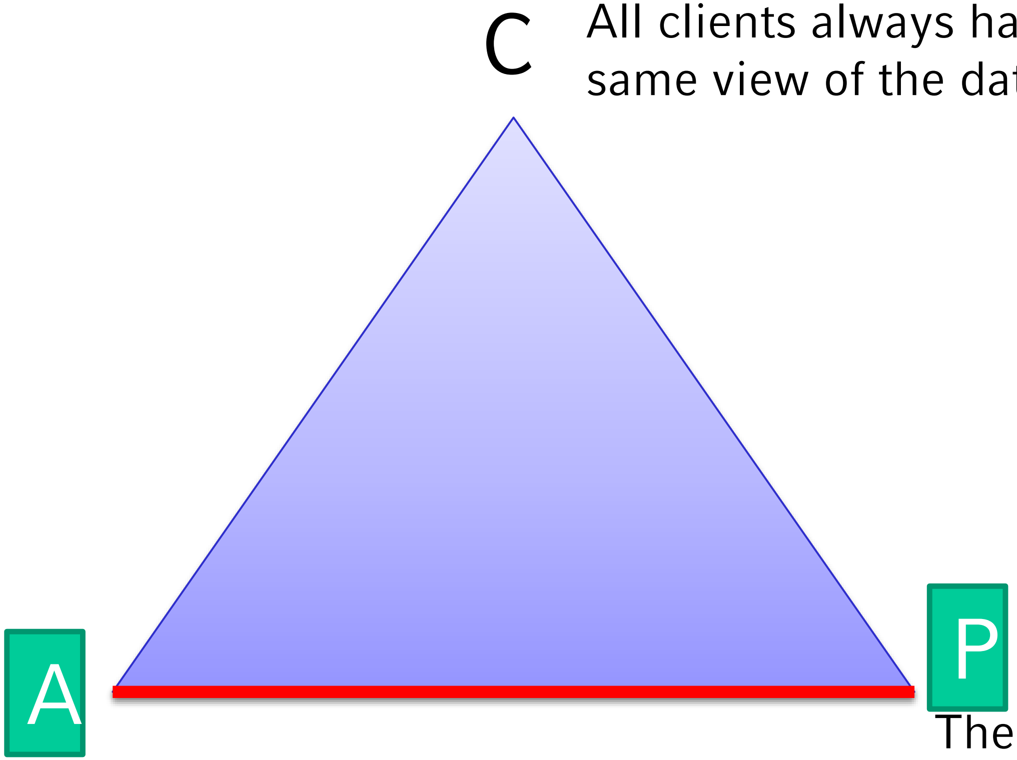
→ In distributed systems, only two of these requirements can be fullfilled at the same time

# Assignment 2-1

- Determine for the following database systems which of the requirements are fulfilled:

# Assignment 2-1

(a) At a cloud-service provider the data is distributed among many, but not fail-safe nodes. They shall be always available for the customer



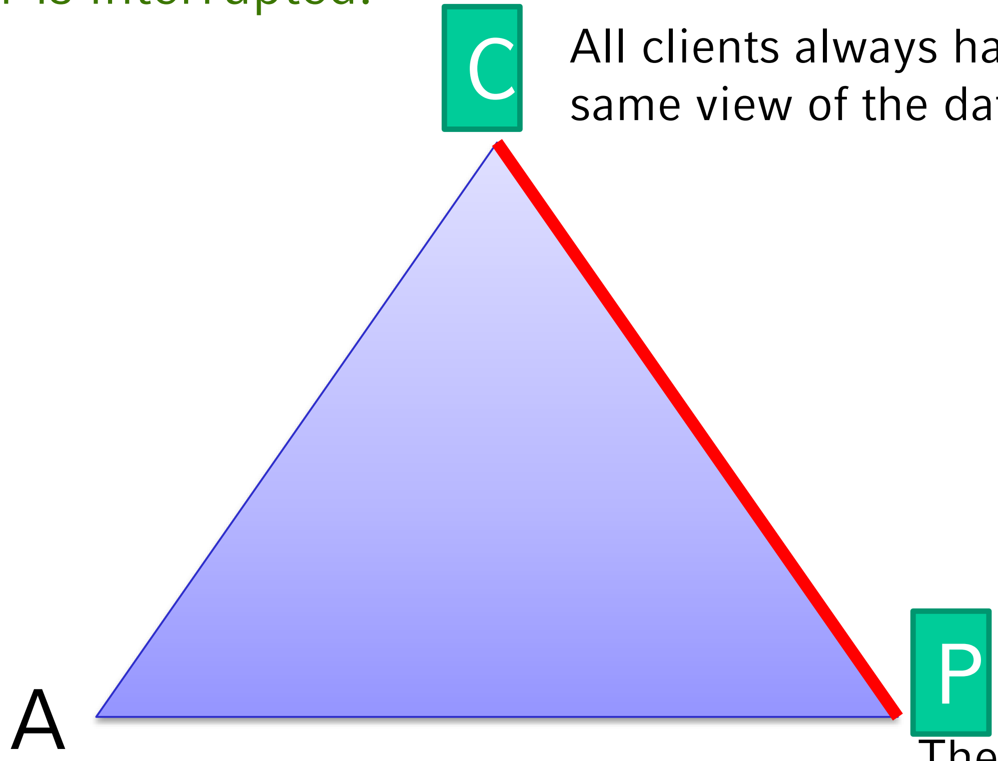
All clients always have the same view of the data

Each client can always read and write

The system works well despite physical network partitions

# Assignment 2-1

(b) A larger bank has per city several ATMs. At the ATMs each of the performed transactions must be stored accurately on the database, even if the data transfer is interrupted.



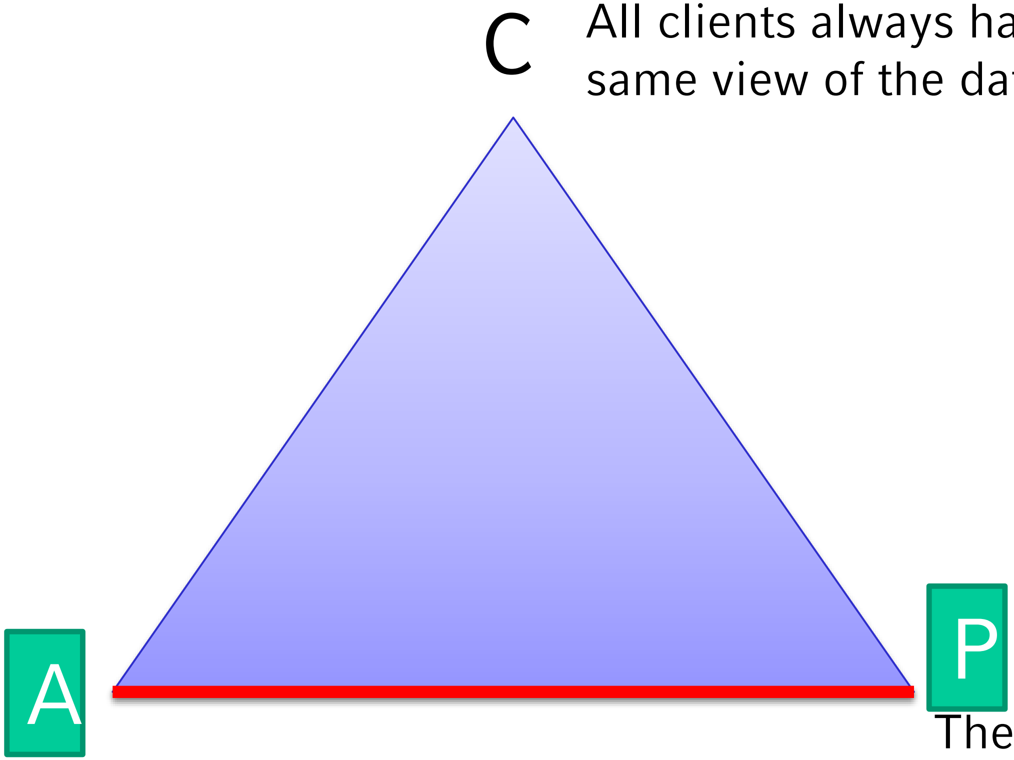
All clients always have the same view of the data

The system works well despite physical network partitions

Each client can always read and write

# Assignment 2-1

(c) Given a social network. It is important that once submitted posts are not again deleted. Further the posts shall be accessible by its users.



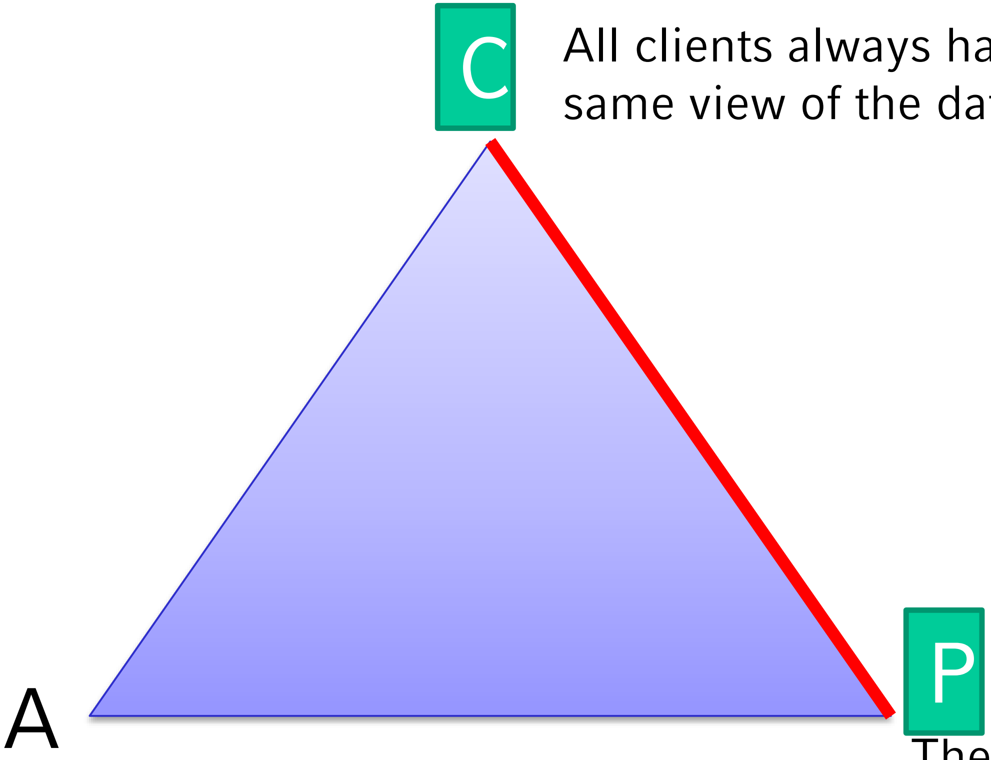
All clients always have the same view of the data

Each client can always read and write

The system works well despite physical network partitions

# Assignment 2-1

(d) On the website of a hotel, the customers can book a room online. Double bookings shall be excluded, as far as possible.



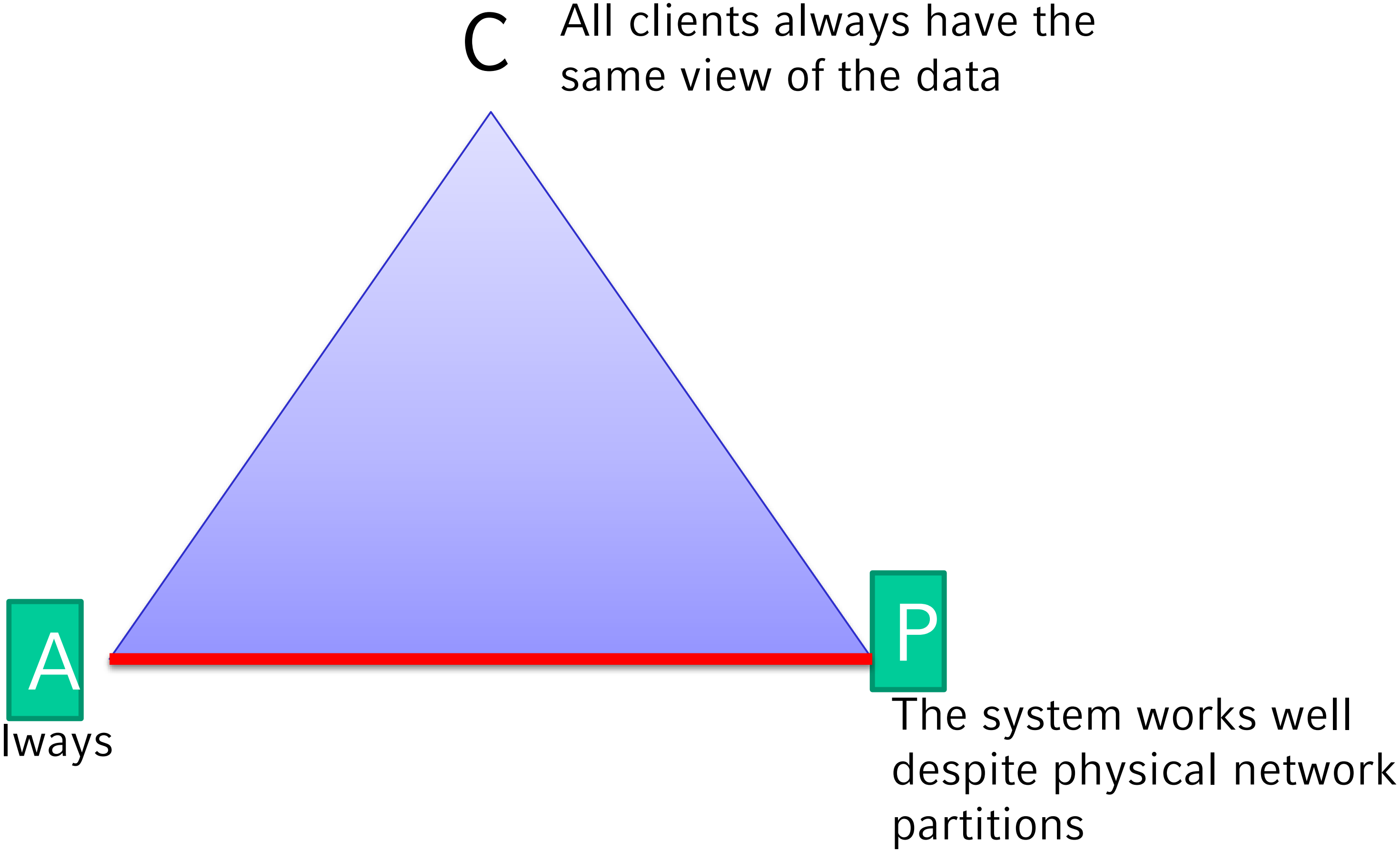
All clients always have the same view of the data

Each client can always read and write

The system works well despite physical network partitions

# Assignment 2-1

(e) A news agency publishes on its website articles on a regular basis. The website shall be always kept up-to-date, as far as possible.


C
All clients always have the same view of the data
A
Each client can always read and write
P
The system works well despite physical network partitions

# Assignment 2-2

## (a) RDMS vs. document database systems

| RDBMS | MongoDB |
|---|---|
| Database | Database |
| Table | Collection |
| Row | Document |
| Index | Index |
| Join | Embedded Document or reference |

# Assignment 2-2

## (c) Create a database named mydb

```
use mydb
```

# Assignment 2-2

(d) Add to the database a collection named city by inserting first a document with the following fields and ist corresponding values: The city name is Munich, its population is 1450381, it is famous for a list of things, such as e.g. the Frauenkirche, the LMU and the Oktoberfest. The document has another document in it, which describes a mayor. The mayor document has a field of ist name Dieter Reiter and the party he belongs to which is the SPD.

```
db.city.insert({name : „Munich“,
                population : 1450381,
                famous_for : [„Frauenkirche“, „LMU“, „Oktoberfest“],
                mayor : { name : „Dieter Reiter“
                          party : „SPD“} })
```

# Assignment 2-2

(e) Add to the city collection another place. The city name is Berlin, its population is 3520031. It is famous for a list of things such as the Brandenburger Tor, Reichstag building, and Charlottenburg. It also has a mayor document with the mayor name Michael Müller and the party he belongs to (SPD).

```
db.city.insert({name : „Berlin“,
               population : 3520031,
               famous_for : [„Brandenburger Tor“, „Reichstag“, „Charlottenburg“],
               mayor : { name : „Michael Müller“
                        party : „SPD“} })
```

# Assignment 2-2

(f) Search in your city collection for a place which has a population larger than three million.

```
db.city.find({„population“ : {$gt : 3000000}})
```

# Assignment 2-2

(g) Search for all cities whose mayor are from the SPD party

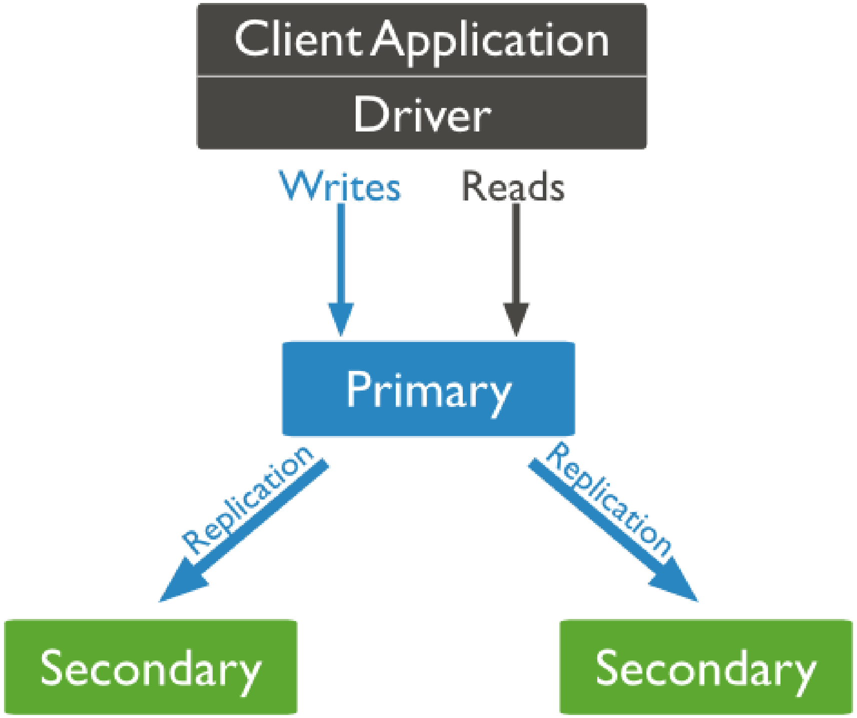
```
db.city.find({„mayor.party“ : „SPD“})
```

# Assignment 2-2

(h) What is a Replication in MongoDB?

- Data is being copied from primary nodes to secondary nodes in order to have a backup copy.
- If the primary node fails, a secondary node is being elected as a new primary

→ It is advised to always provide an odd number of nodes


Client Application
Driver
Writes
Reads
Primary
Replication
Replication
Secondary
Secondary