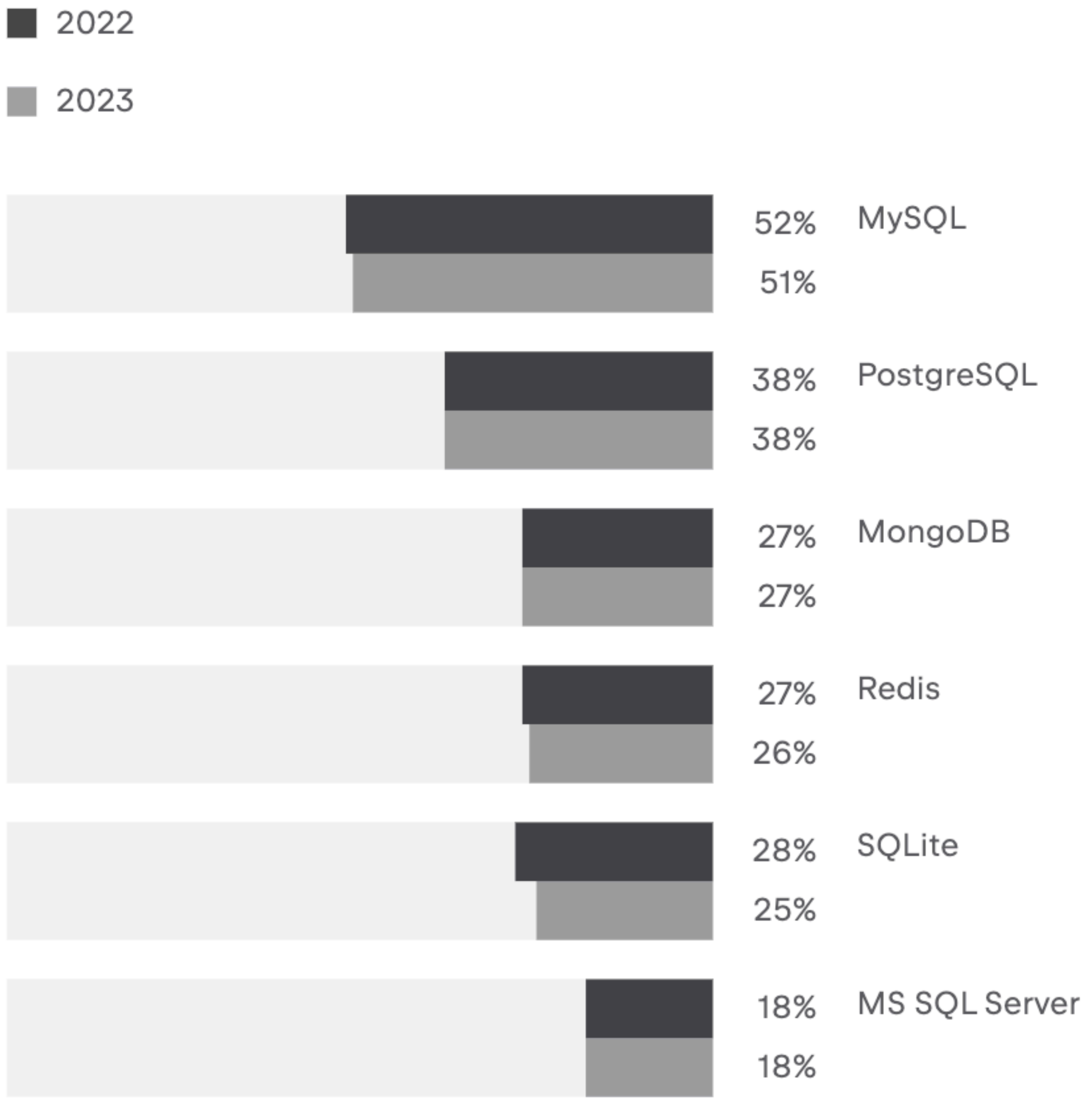
Introducing MongoDB

NoSQL og DBMS

NIFR 2024

Which databases have you used in the last 12 months?



MySQL	PostgreSQL	MongoDB	Redis	SQLite	MS SQL Server	MariaDB	Oracle Database	
-18%	-46%	-27%	-48%	16%	-36%	-29%	-32%	C
-25%	-9%	-7%	-24%	23%	203%	-15%	-17%	C#
-17%	-38%	-19%	-46%	13%	-22%	-35%	-22%	C++
-2%	5%	40%	-14%	73%	-17%	20%	-35%	Dart
26%	48%	39%	126%	32%	-51%	20%	-37%	Go
24%	5%	11%	26%	-19%	-9%	-6%	82%	Java
19%	6%	40%	7%	-2%	0%	31%	-5%	JavaScript
-14%	8%	3%	-10%	66%	-41%	-14%	-21%	Kotlin
68%	0%	-9%	44%	21%	-15%	211%	-48%	PHP
0%	12%	10%	-9%	29%	-10%	-5%	-7%	Python
16%	70%	-37%	82%	17%	-29%	12%	-37%	Ruby
-26%	45%	10%	42%	79%	-53%	11%	-52%	Rust
20%	25%	-5%	24%	-6%	79%	20%	87%	SQL
-41%	67%	-21%	13%	-41%	-49%	-45%	-17%	Scala
-23%	-28%	-16%	-39%	67%	-40%	-31%	-66%	Swift
-7%	47%	48%	24%	9%	-3%	19%	-34%	TypeScript

-66% 0 211%

https://www.jetbrains.com/lp/devecosystem-2023/databases/

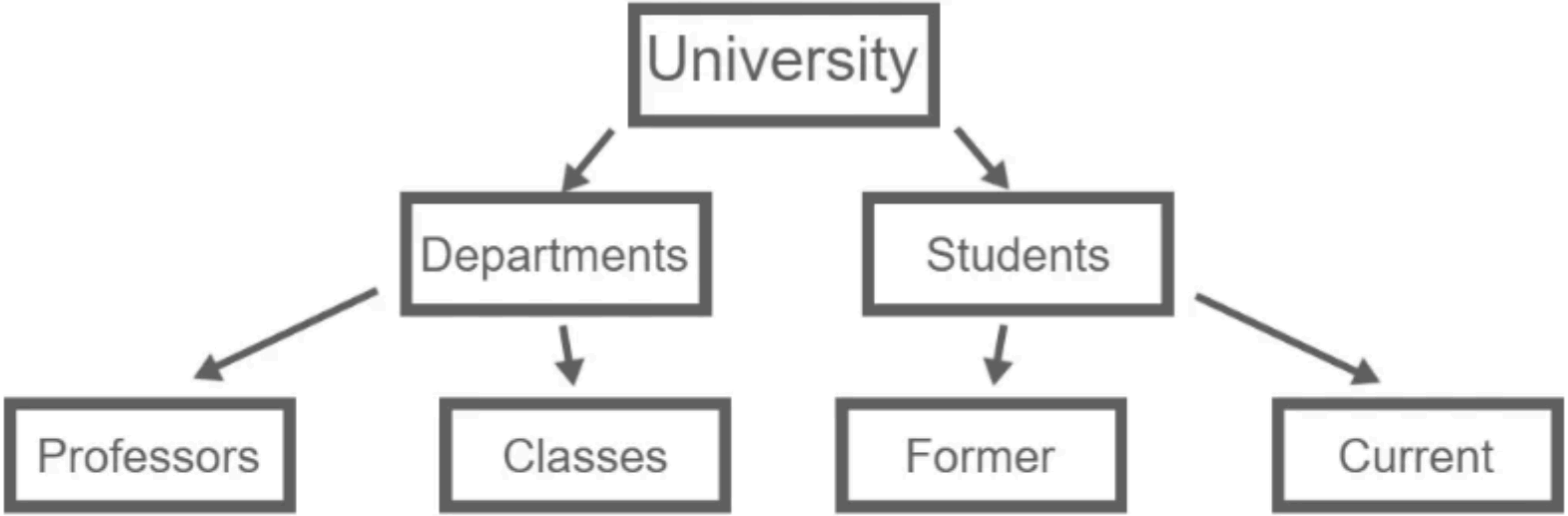
423 systems in ranking, October 2024

Rank			DBMS	Database Model	Score		
Oct 2024	Sep 2024	Oct 2023			Oct 2024	Sep 2024	Oct 2023
1.	1.	1.	Oracle	Relational, Multi-model	1309.45	+22.85	+48.03
2.	2.	2.	MySQL	Relational, Multi-model	1022.76	-6.73	-110.56
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	802.09	-5.67	-94.79
4.	4.	4.	PostgreSQL	Relational, Multi-model	652.16	+7.80	+13.34
5.	5.	5.	MongoDB	Document, Multi-model	405.21	-5.02	-26.21
6.	6.	6.	Redis	Key-value, Multi-model	149.63	+0.20	-13.33
7.	7.	↑ 11.	Snowflake	Relational	140.60	+6.88	+17.36
8.	8.	↓ 7.	Elasticsearch	Multi-model	131.85	+3.06	-5.30
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model	122.77	-0.28	-12.10
10.	10.	↓ 9.	SQLite	Relational	101.91	-1.43	-23.23

Formål og læringsmål

Fagelementet databasedesign indeholder hvordan man ud fra en datamodel implementerer og vedligeholder en relationel database og sætter data ind i denne på en sikker måde. Derudover indeholder fagelementet alternativer til relationelle databaser og analyseret hvilken databaseteknologi der passer bedst til en given opgave.

Hierarchical Data Model 1960's
(Top to bottom / One to many)

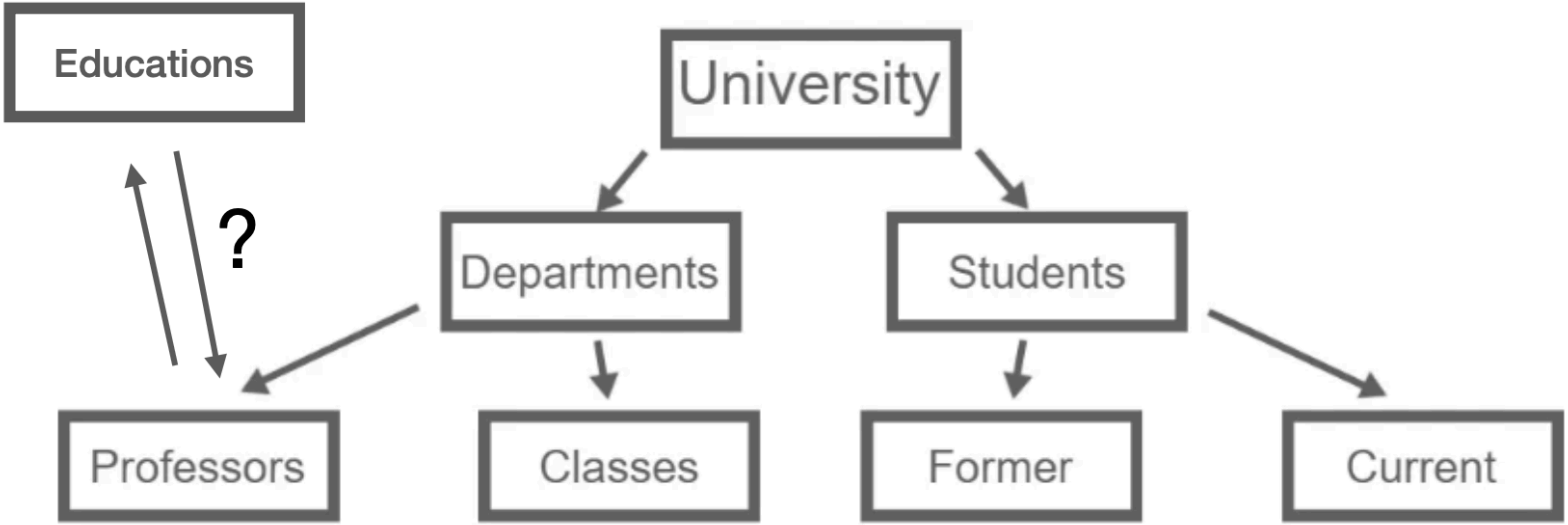


Abstraction: Data is a hierarchy of connected entities from parent to child

```
RD  sales-report
    PAGE LIMITS 60 LINES
    FIRST DETAIL 3
    CONTROLS seller-name.

01  TYPE PAGE HEADING.
    03  COL 1                    VALUE "Sales Report".
    03  COL 74                   VALUE "Page".
    03  COL 79                   PIC Z9 SOURCE PAGE-COUNTER.

01  sales-on-day TYPE DETAIL, LINE + 1.
    03  COL 3                    VALUE "Sales on".
    03  COL 12                   PIC 99/99/9999 SOURCE sales-date.
    03  COL 21                   VALUE "were".
    03  COL 26                   PIC $$$$9.99 SOURCE sales-amount.

01  invalid-sales TYPE DETAIL, LINE + 1.
    03  COL 3                    VALUE "INVALID RECORD:".
    03  COL 19                   PIC X(34) SOURCE sales-record.

01  TYPE CONTROL HEADING seller-name, LINE + 2.
    03  COL 1                    VALUE "Seller:".
    03  COL 9                    PIC X(30) SOURCE seller-name.
```

COBOL language

Hierarchical Data Model 1960's



The relational model 1970

2.4. Summary

In Section 1 a relational model of data is proposed as a basis for protecting users of formatted data systems from the potentially disruptive changes in data representation caused by growth in the data bank and changes in traffic.

In Section 2 operations on relations and two types of redundancy are defined and applied to the problem of maintaining the data in a consistent state. This is bound to become a serious practical problem as more and more different types of data are integrated together into common data banks.

Existing non-inferential, formatted data systems provide users with tree-structured files or slightly more general network models of the data. In Section 1, inadequacies of these models are discussed. A model based on *n*-ary relations, a normal form for data base relations, and the concept of a universal data sublanguage are introduced. In Section 2, certain operations on relations (other than logical inference) are discussed and applied to the problems of redundancy and consistency in the user's model.

https://dl.acm.org/doi/10.1145/362384.362685

TL:DR

E.F Codds analysis

- Tree or network models are inadequate
- Data models are often subject to change
- A model based on relations offers more flexibility
- In addition - a common query language would be practical (SQL)

Abstraction: Data is a set of entities with interconnected relationships

What is the point?

Brief history of databases

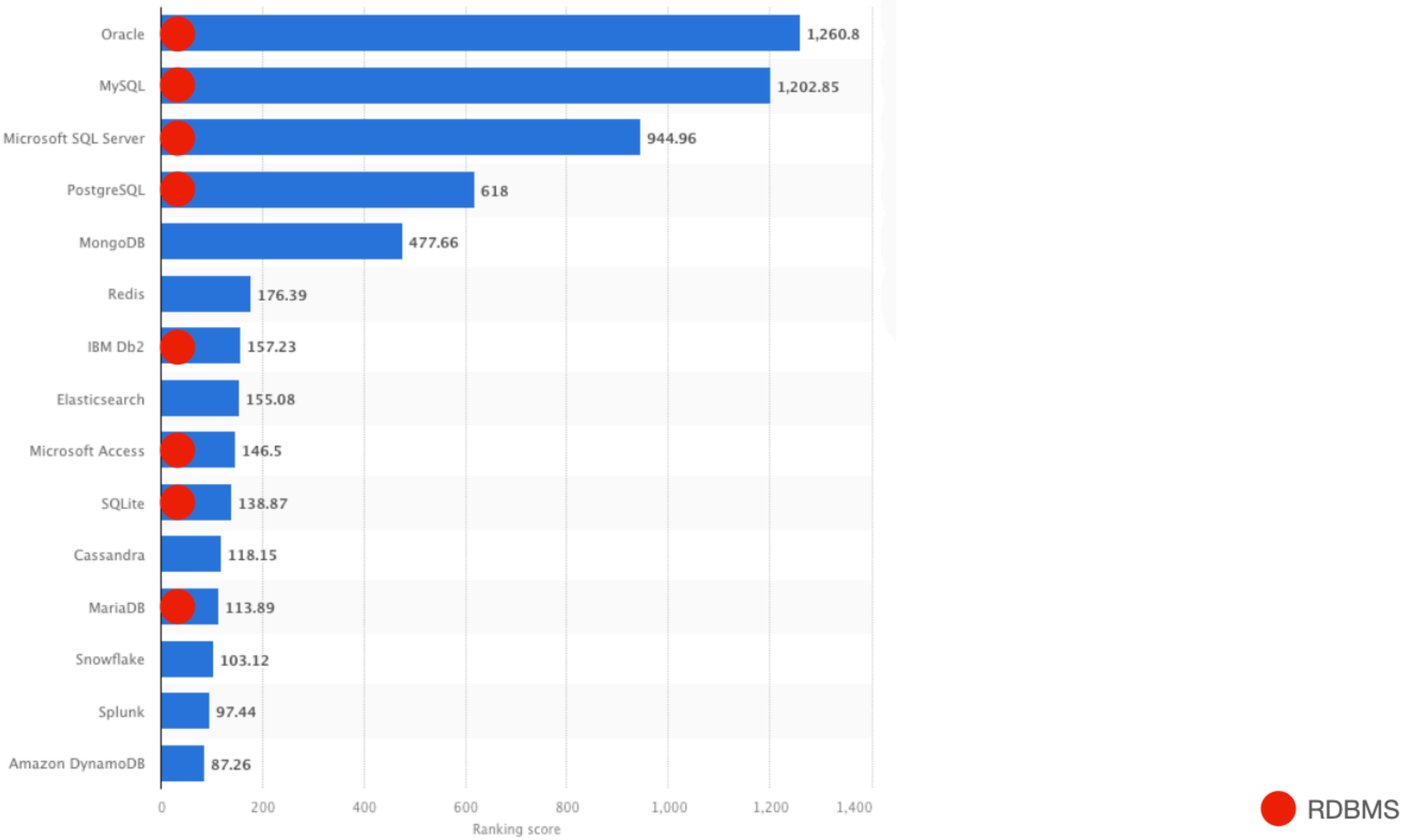
- Abstractions are the essential characteristics of a system
- Abstractions lies at the heart of any implementation of a DBMS
- Technologies represents several implementations of a data storage abstraction
- RDBMS:

Relational Databases are (still) dominating the landscape

Supplementary notes

According to the source, the ranking is a measure of the following parameters:

- website mentions;
- search frequency;
- technical discussion frequency;
- current job offers;
- professional network profiles;
- social network relevance.



https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/#:~:text=As%20of%20August%202022%2C%20the,rounded%20out%20the%20top%20three.

The Cap Theorem

What do we want from a database?

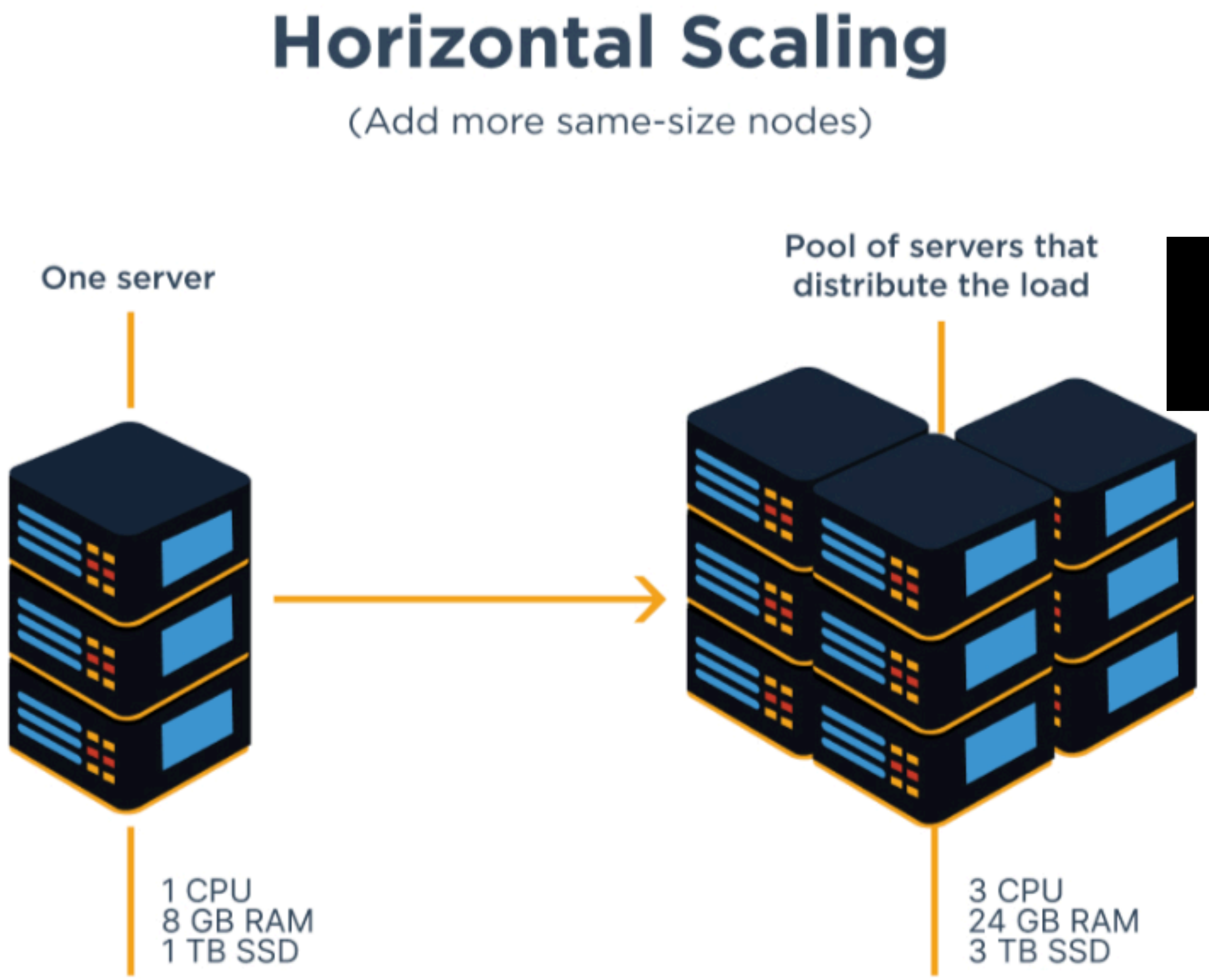
Consistency

Availability

Partition Tolerence

The Cap Theorem

Partitioning

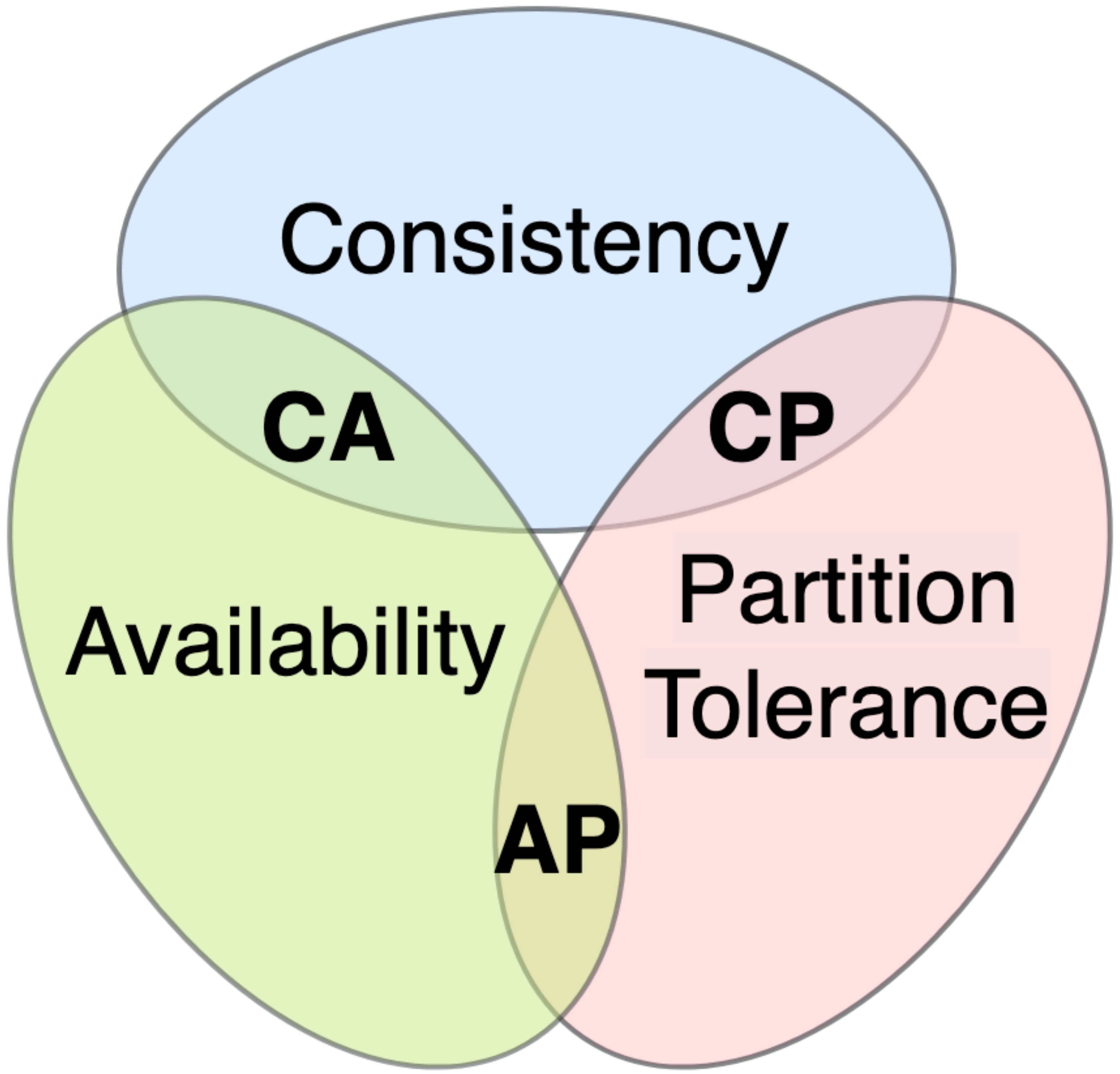


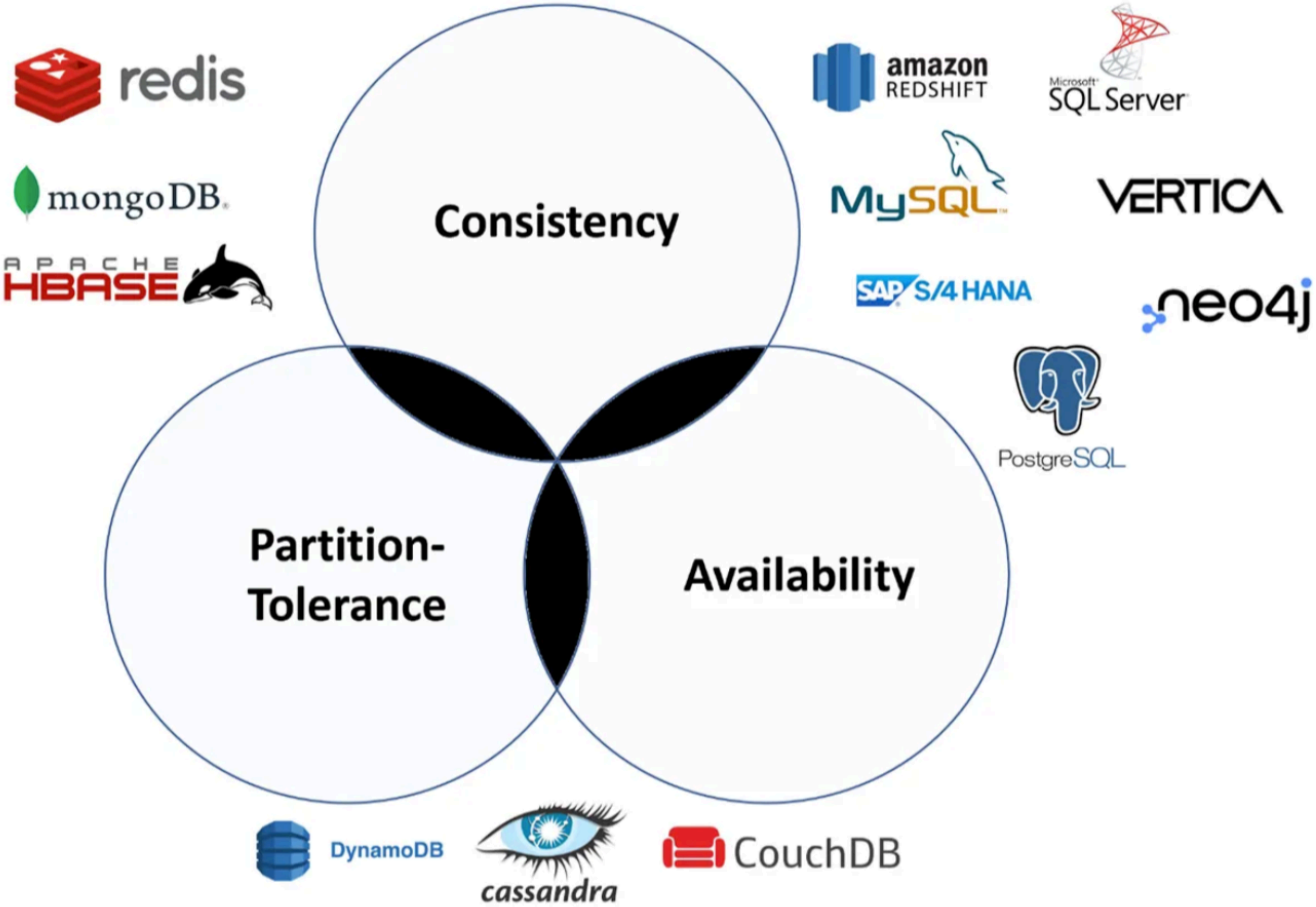
Data is partitioned

The Cap Theorem

Pick 2 - not 3

- Consistency: Every read reflects the most recent write. When you update some data, every user or system trying to access data should see the latest version
- Availability: Every request gets a response (success or failure), but it does not guarantee that the data is up-to-date.
- Partition tolerence: The database continues to operate even if communication between parts of the system is lost (if the network fails)



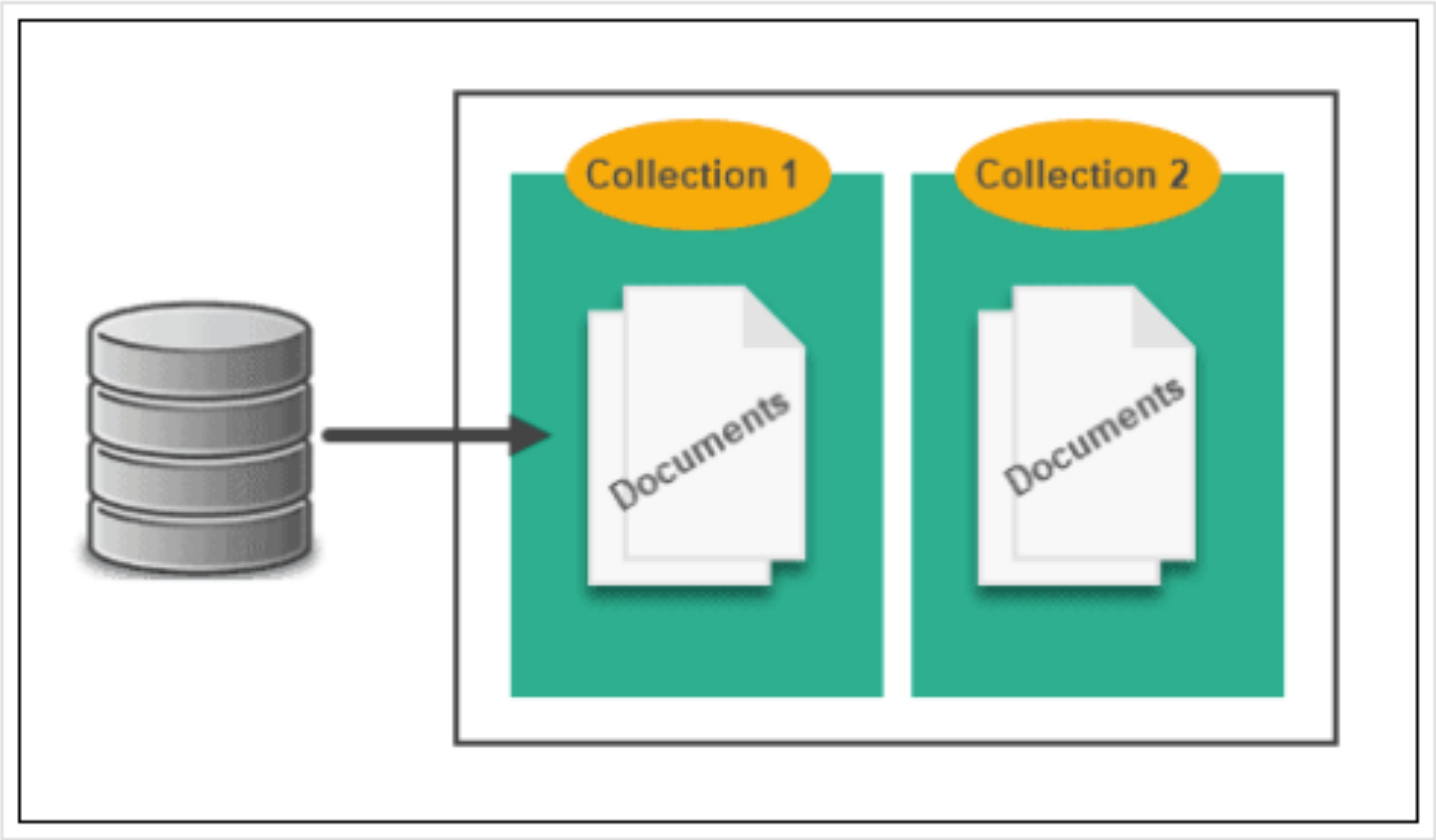


Property 1: Consistency

NoSQL

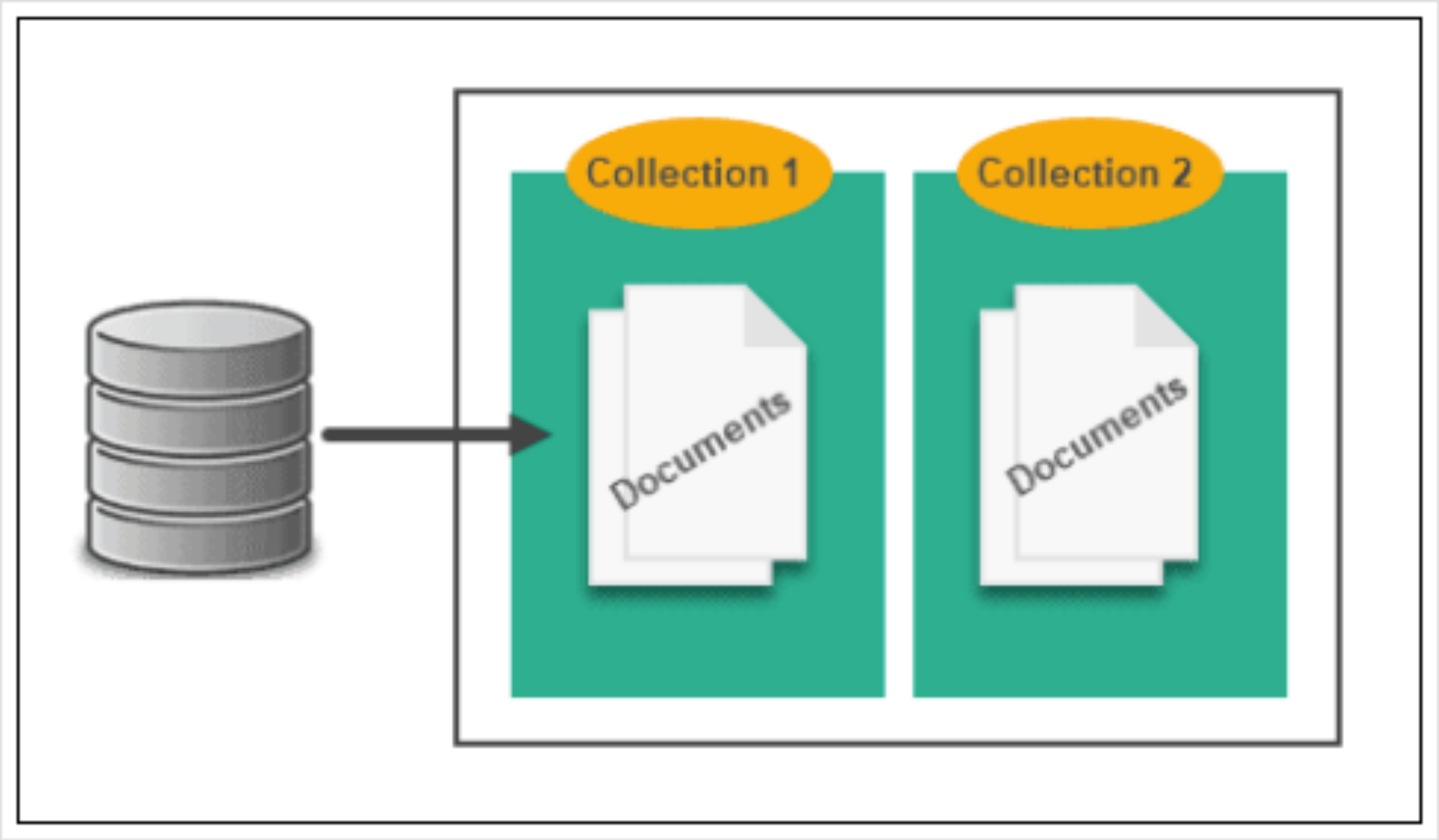
Database types

- Document databases
- Key-Value store databases
- Wide-column stores
- Graph databases



NoSQL

Document database abstraction

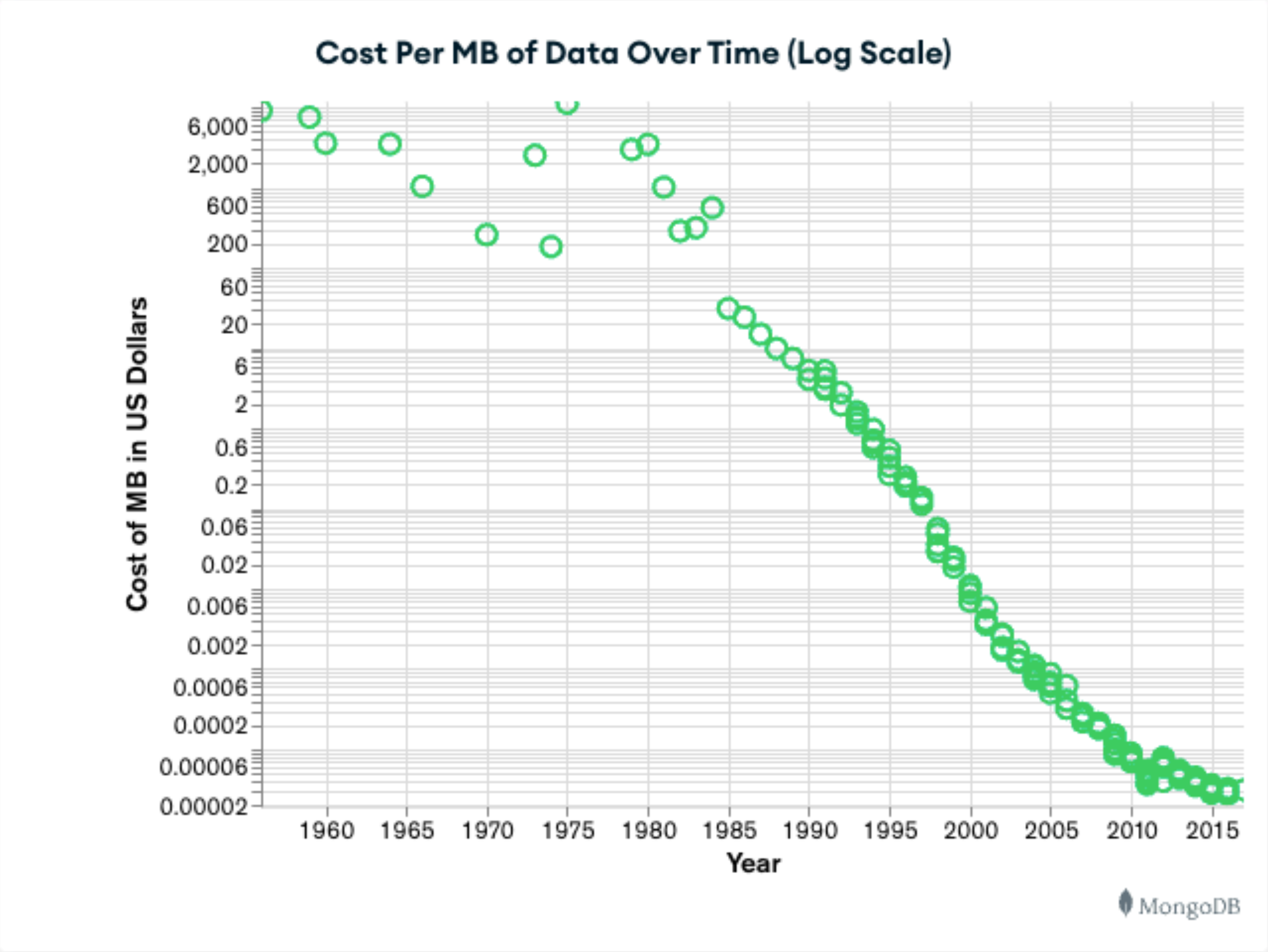



```json
{
  "_id": "12345",
  "name": "foo bar",
  "email": "foo@bar.com",
  "address": {
    "street": "123 foo street",
    "city": "some city",
    "state": "some state",
    "zip": "123456"
  },
  "hobbies": ["music", "guitar", "reading"]
}
```

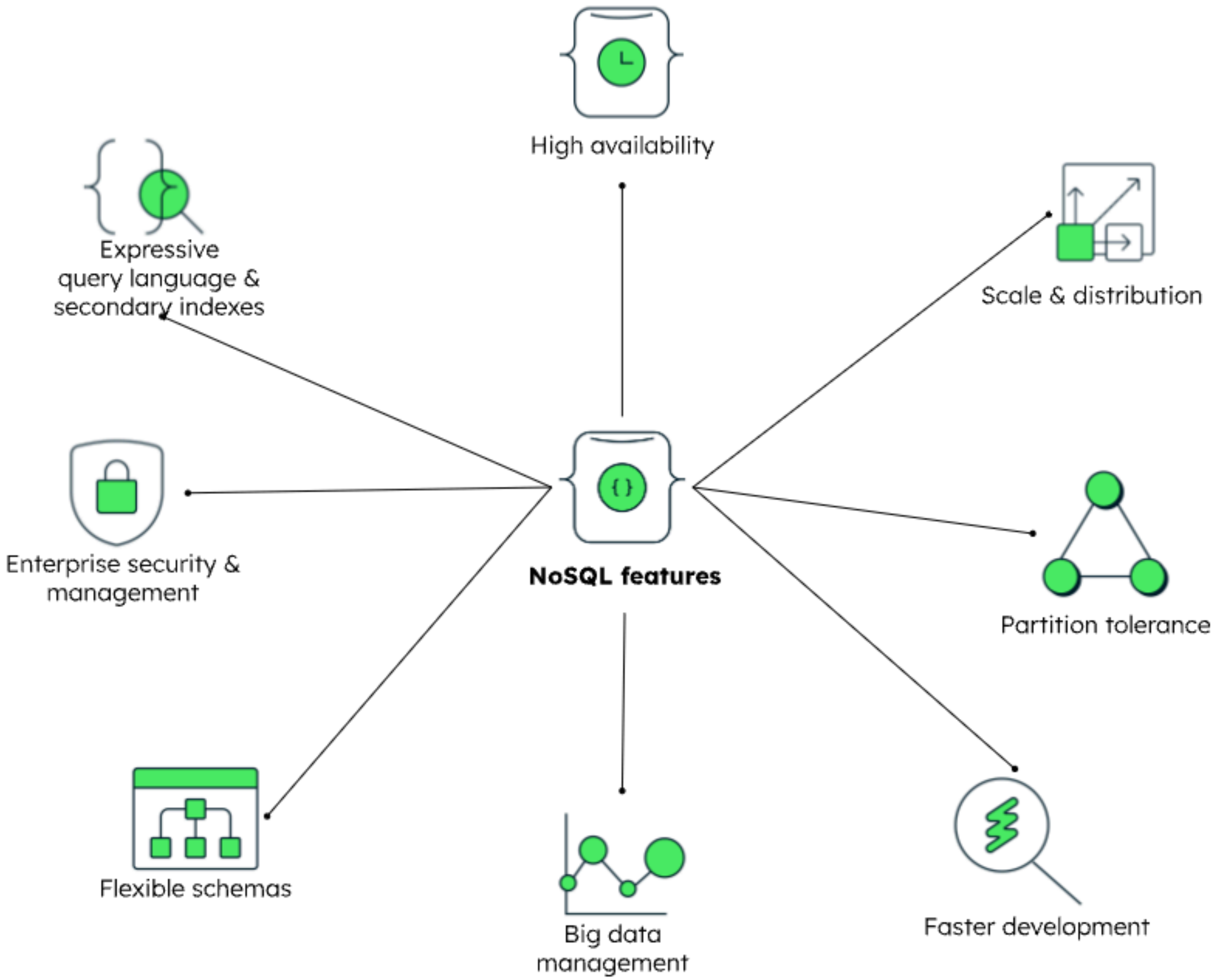
NoSQL

Relevance & emergence

- NoSQL databases emerged in the late 2000s as the cost of storage dramatically decreased. Gone were the days of needing to create a complex, difficult-to-manage data model in order to avoid data duplication. NoSQL databases optimized for developer productivity.



https://www.mongodb.com/resources/basics/databases/nosql-explained


High availability
Expressive
query language &
secondary indexes
Scale & distribution
Enterprise security &
management
NoSQL features
Partition tolerance
Flexible schemas
Big data
management
Faster development

RDBMS vs NoSQL (Document)

Relational Database

User table

ID	first_name	last_name	cell	city
1	Leslie	Yepp	8125552344	Pawnee

Hobbies table

ID	user_id	hobby
10	1	scrapbooking
11	1	eating waffles
12	1	working

MongoDB

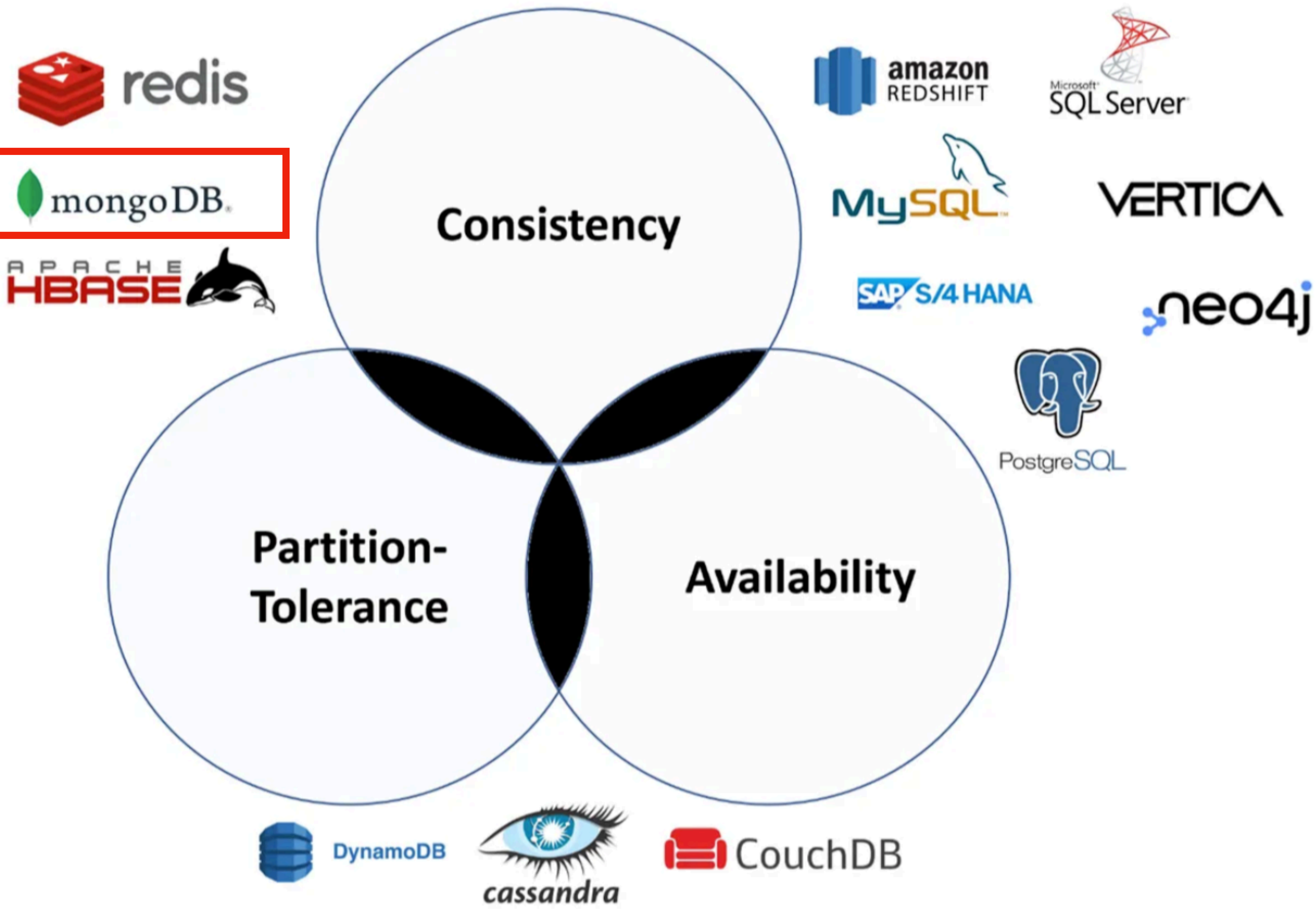
```
{
   "_id": 1,
   "first_name": "Leslie",
   "last_name": "Yepp",
   "cell": "8125552344",
   "city": "Pawnee",
   "hobbies": ["scrapbooking", "eating
waffles", "working"]
}
```

- No need for joins
- No need for data normalization

When to use MongoDB?

Vs. MySQL

- Storage of unstructured and semi-structured data
- Huge volumes of data
- Requirements for scale-out architecture
- Modern application paradigms like microservice and real time streaming
- High write load
- Geospatial data
- Document-based data

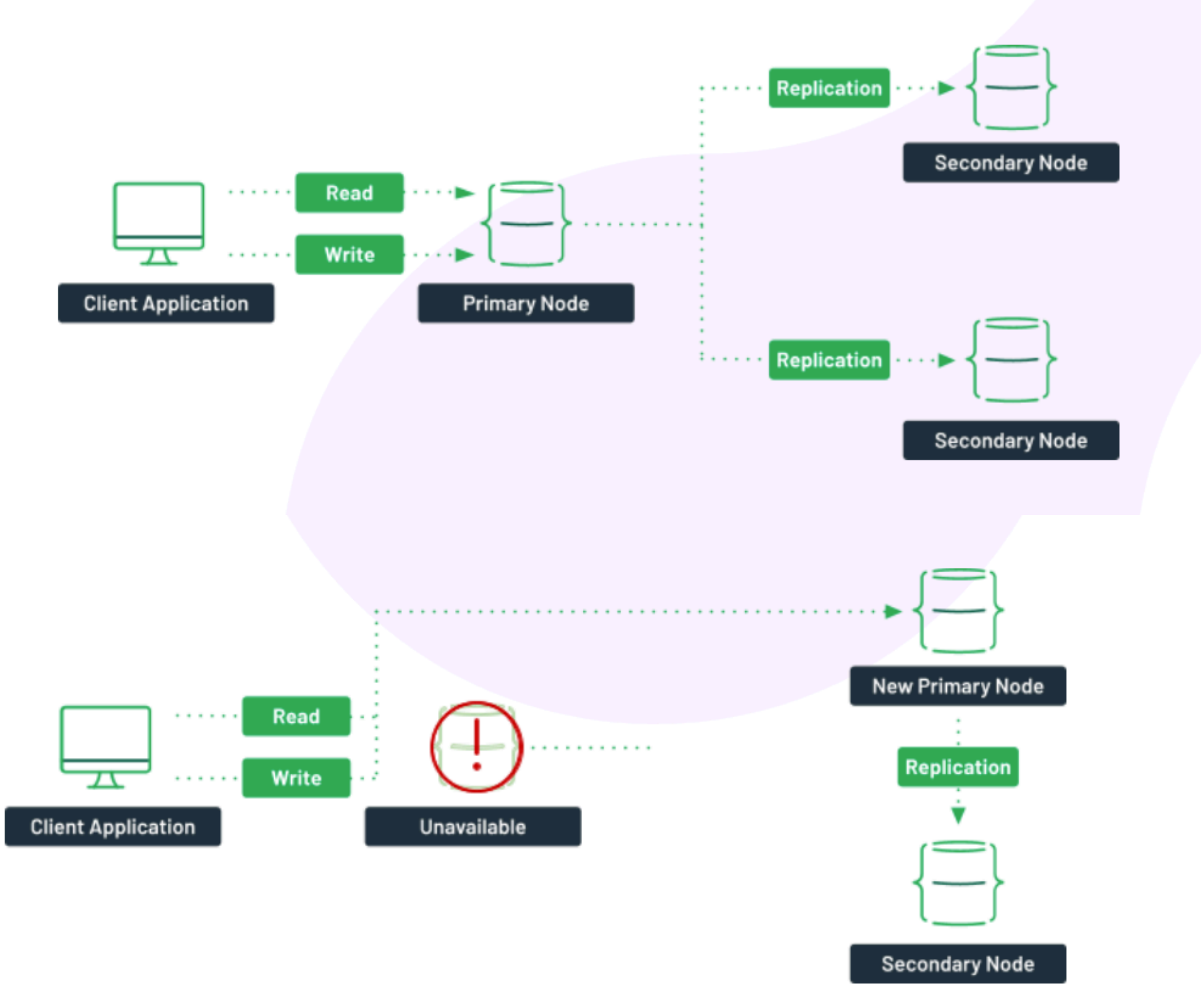


Property 1: Consistency

MongoDB

Sacrifices availability

Split Brain Scenario



Installing MongoDB compass

Finding fire Pokemon with a base_happiness > 50

Queries

Or Operator

```
{
  $or: [
    { "type1": "water" },
    { "type1": "fire" }
  ]
}
```

Queries

And & OR Operator

```
$and: [
    {
      $or: [
        { "type1": "grass" },
        { "type1": "electric" }
      ]
    },
    { "speed": { $gt: 80 } }
  ]
}
```

Queries

Less than

```
{
  "base_total": { $lt: 300 }
}
```

Queries

Regular Expressions

```
{
  $or: [
    // Names starting with "S" (case-insensitive)
    { "name": { $regex: "^S", $options: "i" } },
     // Names ending with "e" (case-insensitive)
    { "name": { $regex: "e$", $options: "i" } }
  ]
}
```

Spørgetime: Tid til at finde spørgsmål