Introduction to NoSQL databases

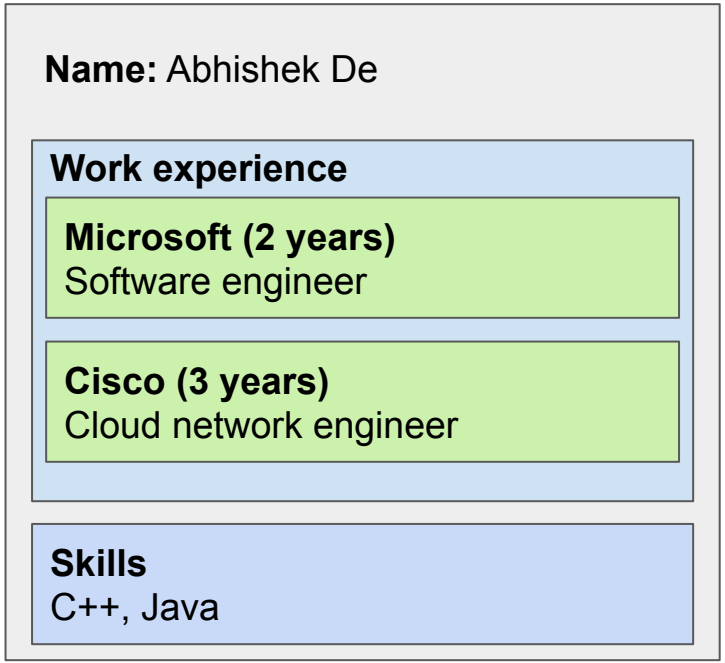
Name: ABHISHEK DE
Roll no: 001710501027
BCSE - IV

A brief recap of Relational Databases

- Data is stored in **tables** that have fixed **rows and columns**.
- Databases are accessed by **SQL(Structured Query Language)**
- Rose in popularity in early 1970s, when storage was very expensive.
- Hence in order to reduce **data duplication**, data was split into multiple tables.
- To retrieve data split across multiple tables, **JOIN** operations was used.
- Also at that time, the waterfall model was followed, hence the requirements were generally freezed before development and database design was a one time and carefully planned operation.

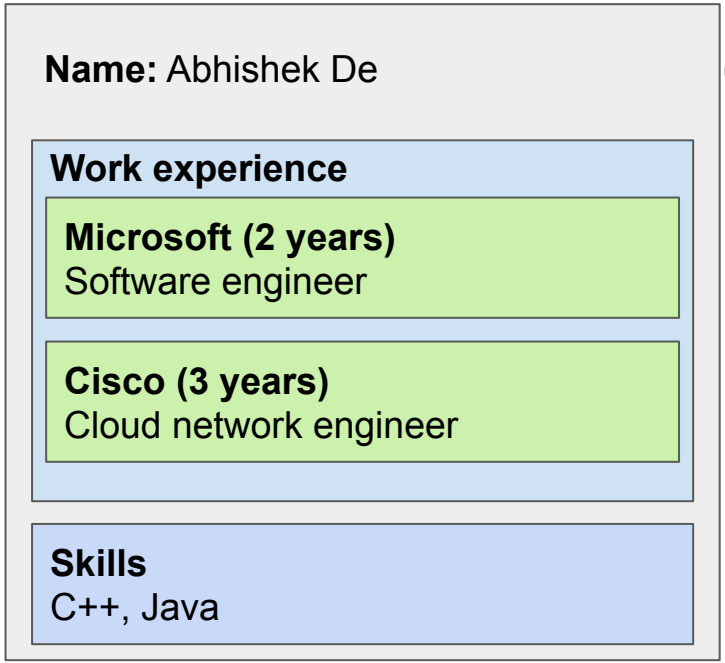
Some drawbacks of relational databases.

- Imagine storing a resume in a relational database.



Some drawbacks of relational databases.

- Imagine storing a resume in a relational database.

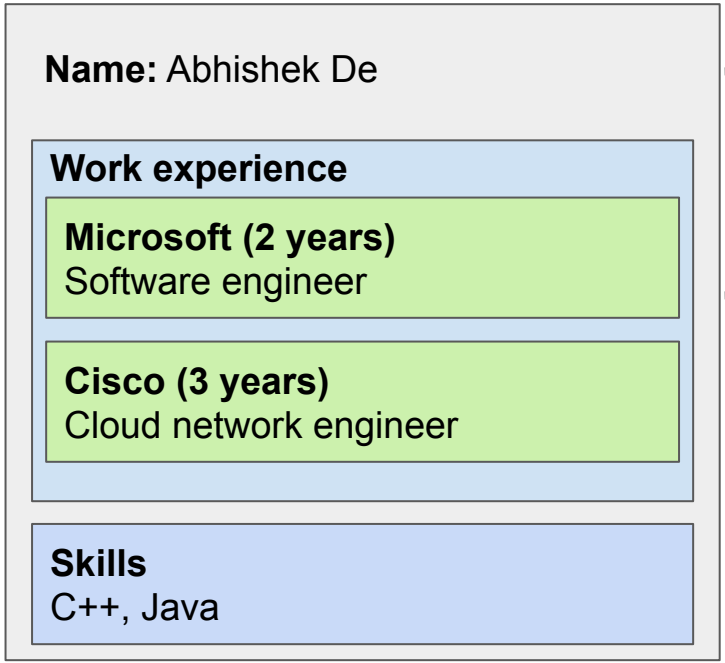


ID	First	Last
1	Abhishek	De

APPLICANT

Some drawbacks of relational databases.

- Imagine storing a resume in a relational database.



APPLICANT

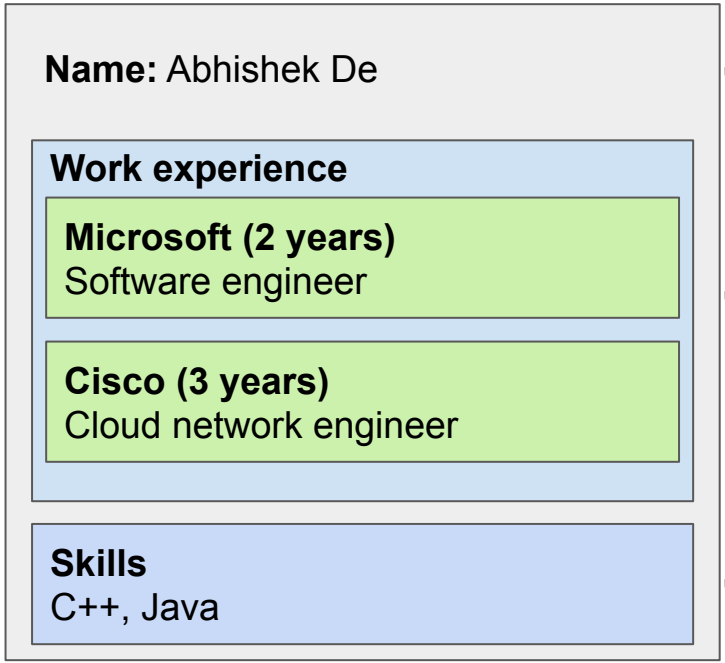
ID	First	Last
1	Abhishek	De

EXPERIENCE

ID	Company	Years	Role
1	Microsoft	2	…...
1	Cisco	3	…...

Some drawbacks of relational databases.

- Imagine storing a resume in a relational database.



APPLICANT

ID	First	Last
1	Abhishek	De

EXPERIENCE

ID	Company	Years	Role
1	Microsoft	2	…...
1	Cisco	3	…...

SKILL

ID	Skill
1	C++
1	Java

Some drawbacks of relational databases.

- Now suppose we want the name of applicants who have worked as a software engineer and are skilled in Java.

APPLICANT

ID	First	Last
1	Abhishek	De

EXPERIENCE

ID	Company	Years	Role
1	Microsoft	2	.....
1	Cisco	3	.....

SKILL

ID	Skill
1	C++
1	Java

Some drawbacks of relational databases.

- Now suppose we want the name of applicants who have worked as a software engineer and are skilled in Java.

APPLICANT

ID	First	Last
1	Abhishek	De

EXPERIENCE

ID	Company	Years	Role
1	Microsoft	2	……
1	Cisco	3	……

SKILL

ID	Skill
1	C++
1	Java

```
SELECT FIRST, LAST
FROM ((APPLICANT
INNER JOIN EXPERIENCE
ON APPLICANT.ID = EXPERIENCE.ID)
INNER JOIN SKILL
ON APPLICANT.ID = SKILL.ID)
WHERE SKILL = ‘Java’
AND ROLE = ‘Software Engineer’;
```

JOINS ARE EXPENSIVE!

Some drawbacks of relational databases.

- Suppose we need to add a new attribute '**Email ID**' to our previous table.

ID	First	Last
1	Abhishek	De

- We cannot insert new data without changing the schema.
- To insert data, developers need to request a schema change from the database administrator.
- This slows down development, not only because this is a manual process, but it impacts other applications and services.

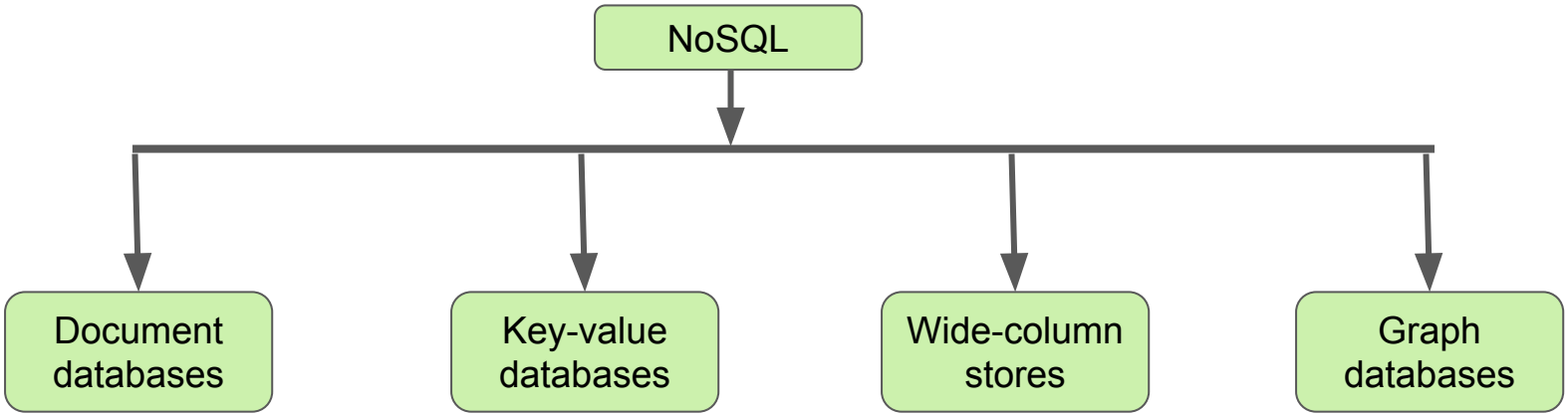
Some more drawbacks…..

- Data type of attributes are also rigid and decided during schema creation.
- The way data is modelled in databases is quite different from how it is used in code.
- Relational databases are not horizontally scalable.

What is NoSQL?

- NoSQL is an **umbrella term** to describe any alternative system to traditional SQL databases.
- They all use a data model that has a different structure than the traditional row-and-column table model used with relational database management systems.
- Although such databases have existed since the late 1960s, the name “NoSQL” was coined only in 21st century, triggered by the needs of big data companies where **speed**, **flexibility**, **scalability** and **ease of development** became important issues.

Types of NoSQL databases



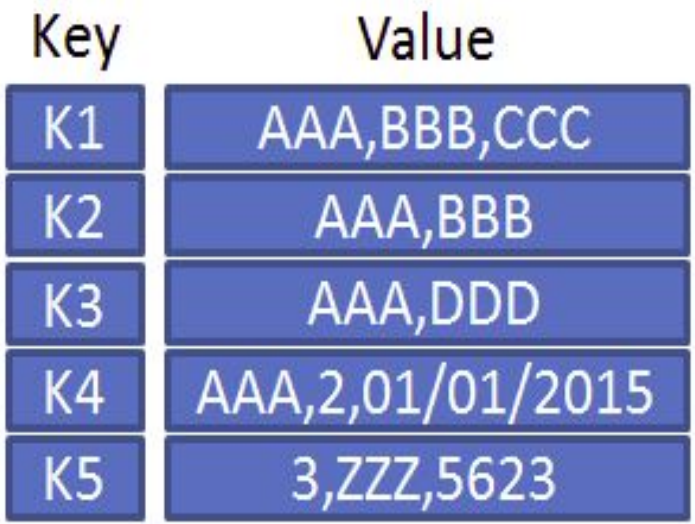
Document databases

- Most popular type of NoSQL database.
- Stores data in JSON, BSON or XML documents.
- Documents can be **nested**. Multivalued and composite attributes can be stored and retrieved efficiently.
- Closer to data objects used in applications, hence easier to use with code.
- **Schemaless**, can be restructured easily.
- Examples: MongoDB, Couch DB, Cosmos DB

```
"customer" =
{
   "id": "Customer:1",
   "firstName": "John",
   "lastName": "Wick",
   "age: 25,
   "address": {
        "country": "US",
        "city": "New York",
        "state": "NY"
        "street": "21 2nd Street",
      },
   "hobbies": [ Football, Hiking ],
   "phoneNumbers": [
      {
        "type": "Home",
        "number": "212 555-1234"
      },
      {
        "type": "Office",
        "number": "616 565-6789"
      }
    ]
}
```

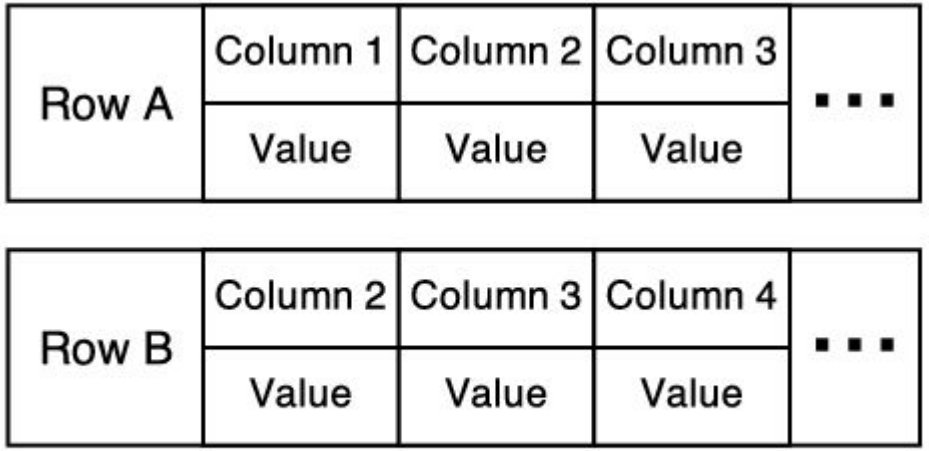
Key-value databases

- Each item is a key-value pair
- Treats the data as a single opaque collection, which may have different fields for every record.
- Optional values are not represented by placeholders, hence far more memory efficient compared to Relational DBs.
- Schemaless, hence provides lot of flexibility.
- Examples: Redis, DynamoDB



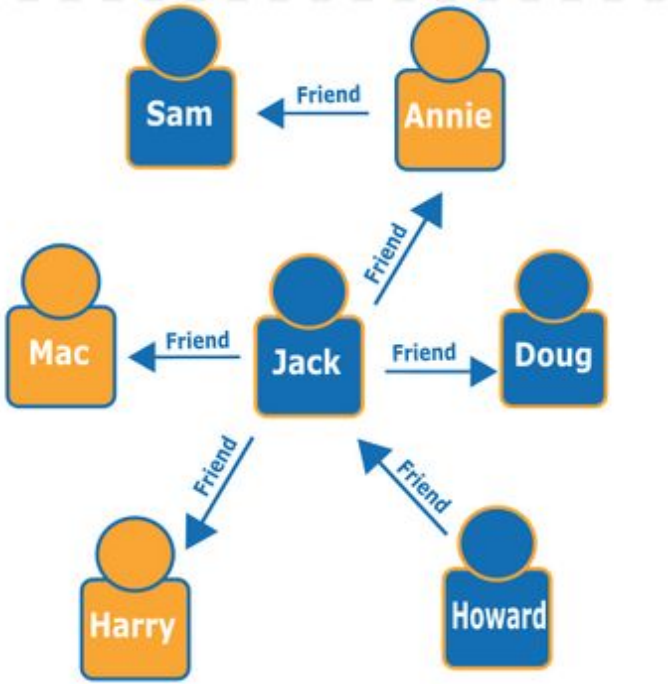
Wide column databases

- Can be interpreted as a 2 dimensional key-value store.
- It uses tables, rows, and columns.
- But unlike a relational database, each row has its **own set of attributes**, which do no span across rows.
- Examples: Cassandra, HBase



Graph databases

- Stores data and relations in the form of nodes and edges.
- Nodes typically store information about entities while edges store information about the relationships between the nodes.
- Much faster than relational databases, as queries can be resolved by following pointers to related nodes rather than performing expensive JOINs.
- Examples: Neo4j, JanusGraph.



General advantages of NoSQL databases

1. **NoSQL is faster since it prevents expensive JOINS.**

General advantages of NoSQL databases

- Getting back to the resume example

Relational DB

APPLICANT

ID	First	Last
1	Abhishek	De

EXPERIENCE

ID	Company	Years	Role
1	Microsoft	2	……
1	Cisco	3	……

SKILL

ID	Skill
1	C++
1	Java

NoSQL DB

```
{
      id: 1,
      first: “Abhishek”,
      last: “De”,
      experience: [
          {
            company: “Microsoft”,
            years: 2,
            role: “Software engineer”,
          },
          {
            company: “Cisco”,
            years: 3,
            role: “Cloud network engineer”,
          }
      ],
      skills: [“C++”, “Java”],
}
```

General advantages of NoSQL databases

- Getting back to the resume example

MySQL command

```
SELECT FIRST, LAST
FROM ((APPLICANT
INNER JOIN EXPERIENCE
ON APPLICANT.ID = EXPERIENCE.ID)
INNER JOIN SKILL
ON APPLICANT.ID = SKILL.ID)
WHERE SKILL = ‘Java’
AND ROLE = ‘Software Engineer’;
```

MongoDB command

```
db.applicant.find({
    skills: “Java”,
    experience.role:
        “Software Engineer”,
}, {
    first: 1,
    last: 1,
})
```

General advantages of NoSQL databases

1. **NoSQL is faster since it prevents expensive JOINS.**
2. **NoSQL supports flexible schema, enables easy and frequent changes to database.**

General advantages of NoSQL databases

- For, example if we want to add ‘Email ID’ as an attribute

MySQL command

```
1.  Change the schema
ALTER TABLE APPLICANT
ADD EMAIL VARCHAR(50);

2.  Perform new insertions
INSERT INTO APPLICANT
VALUES (
    “Rahul”,
    “Kumar”,
    “rhkm@gmail.com”
);
```

MongoDB command

```
db.insertOne({
    id: 2,
    first: “Rahul”,
    last: “Kumar”,
    email: “rhkm@gmail.com”,
`   experience: [{
        company: “Uber”,
        years: 1,
        role: “Software engineer”,
    }],
    skills: [“Python”, “ML”],
})
```

General advantages of NoSQL databases

1. **NoSQL is faster since it prevents expensive JOINS.**
2. **NoSQL supports flexible schema, enables easy and frequent changes to database.**
3. **NoSQL also supports data type flexibility.**

General advantages of NoSQL databases

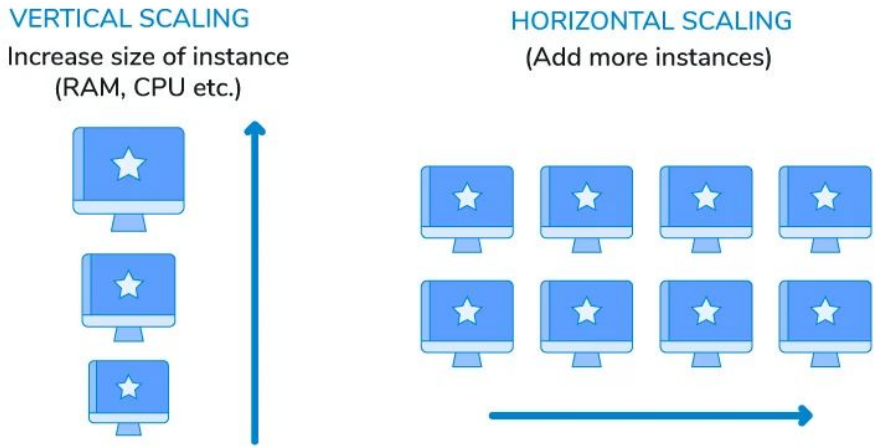
1. **NoSQL is faster since it prevents expensive JOINS.**
2. **NoSQL supports flexible schema, enables easy and frequent changes to database.**
3. **NoSQL also supports data type flexibility.**
4. **NoSQL data models are closer to data objects used in applications, hence more developer friendly.**

General advantages of NoSQL databases

1. **NoSQL is faster since it prevents expensive JOINS.**
2. **NoSQL supports flexible schema, enables easy and frequent changes to database.**
3. **NoSQL also supports data type flexibility.**
4. **NoSQL data models are closer to data objects used in applications, hence more developer friendly.**
5. **NoSQL databases scale well both horizontally and vertically.**

What is scalability and how it applies to databases

- **Scalability** is the ability to adapt a system for handling increasing load.
- **Vertical scalability** refers to ease of increasing the load on a single server by increasing components like RAM, SSD, or CPU.
- **Horizontal scalability** refers to the ease with which we can add more servers to the database.



Scalability of Relational vs NoSQL databases

- NoSQL databases are horizontally scalable due to the following reasons:
  - If relational databases keep related data across multiple servers, accessing them together for JOIN becomes very costly. NoSQL documents are generally **self-contained**, eliminating the need of JOINs, hence horizontally scalable.
  - Relational databases generally follow **ACID** properties, hence if we need to lock two tables for an atomic transaction, and they are on different servers, it is extremely costly. NoSQL generally has document level atomicity, hence such scenarios do not occur.

Disadvantages of NoSQL databases

1. **Most NoSQL databases don’t support ACID properties and might not be a good choice for applications that commit multi-document transactions, for example bank systems.**
2. **NoSQL are generally not compatible with relational databases, hence migration of existing databases might be difficult.**
3. **There is no common, standardized query language as there is in SQL for relational databases.**

References

- https://en.wikipedia.org/wiki/NoSQL
- https://www.mongodb.com/nosql-explained
- https://www.couchbase.com/resources/why-nosql
- https://stackoverflow.com/questions/8729779/why-nosql-is-better-at-scaling-out-than-rdbms
- https://en.wikipedia.org/wiki/Wide-column_store
- https://en.wikipedia.org/wiki/Document-oriented_database
- https://en.wikipedia.org/wiki/Key-value_database
- https://en.wikipedia.org/wiki/Graph_database