NoSQL Database Systems — Unit 7

Dr.-Ing Anna Androvitsanea

IU Internationale Hochschule GmbH

November 28, 2023

STUDY GOALS

- Explain the meaning and the need for NoSQL.
- Discuss the difference between constraints of relational and non-relational data models.
- Explain the differences among different available NoSQL system types.

EXPLAIN SIMPLY

1. List three drawbacks of the relational model.
2. What are the two modern data management needs that led to the adoption of NoSQL paradigms?
3. Many NoSQL systems are open-source. What does open-source mean?

Introduction to NoSQL

What is NoSQL?

NoSQL stands for "Not Only SQL" and refers to a broad class of database management systems that differ from traditional relational database systems in that they do not use SQL as their primary query language.

- Designed to handle large volumes of data
- Can store unstructured, semi-structured, or structured data
- Often used for big data and real-time web applications

WHY NOSQL?

- Release much of the SQL constraints
- Hundreds of thousands of users
- Huge amounts of data
- Flexibility
- Schema-free or weak schema restrictions
- High availability
- Scalability

NOSQL PROPERTIES

- The underlying data model is not relational.
- Distributed and horizontal scalability:
  - Growing data
  - Increasing users
  - Simple replication
  - Semi- and unstructured data

Types of NoSQL Databases

Categories

NoSQL databases can be categorized into four main types:

1. Document-oriented (e.g., MongoDB, CouchDB)
2. Key-value stores (e.g., Redis, DynamoDB)
3. Column-oriented (e.g., Cassandra, HBase)
4. Graph databases (e.g., Neo4j, Amazon Neptune)

Examples of NoSQL Databases

Popular NoSQL Databases

- **MongoDB** - A document-oriented database which is known for its flexibility and ease of use.
- **Redis** - A key-value store that is extremely fast and is often used for caching.
- **Cassandra** - A column-oriented database that excels at handling large datasets distributed across multiple nodes.
- **Neo4j** - A graph database that excels at managing data with complex relationships and network structures.

Advantages of NoSQL

Why NoSQL?

NoSQL databases offer several advantages over traditional relational databases:

- Scalability - Can handle large volumes of data and traffic
- Flexibility - Can store different types of data together
- Performance - Optimized for specific data models and access patterns
- High Availability - Designed for resilience and distributed nature

ACID Properties

Atomicity

Ensures that all operations within a work unit are completed successfully; if not, the transaction is aborted.

Consistency

Ensures that the database properly changes states upon a successfully committed transaction.

Isolation

Ensures that transactions are securely and independently processed at the same time without interference, but it appears to be executed sequentially.

Durability

Guarantees that once a transaction has been committed, it will remain so, even in the case of a system failure.

BASE Properties

Basically Available

Guarantees the availability of the data as much as possible, but does not offer a guarantee of immediate consistency.

Soft state

The state of the system could change over time, even without input.

Eventual Consistency

The system will become consistent over time, given that the system doesn't receive input during that time.

BASE VS. ACID

Comparing BASE VS. ACID highlights the trade-offs between consistency, availability, and partition tolerance in distributed systems, famously formalized in the CAP theorem.

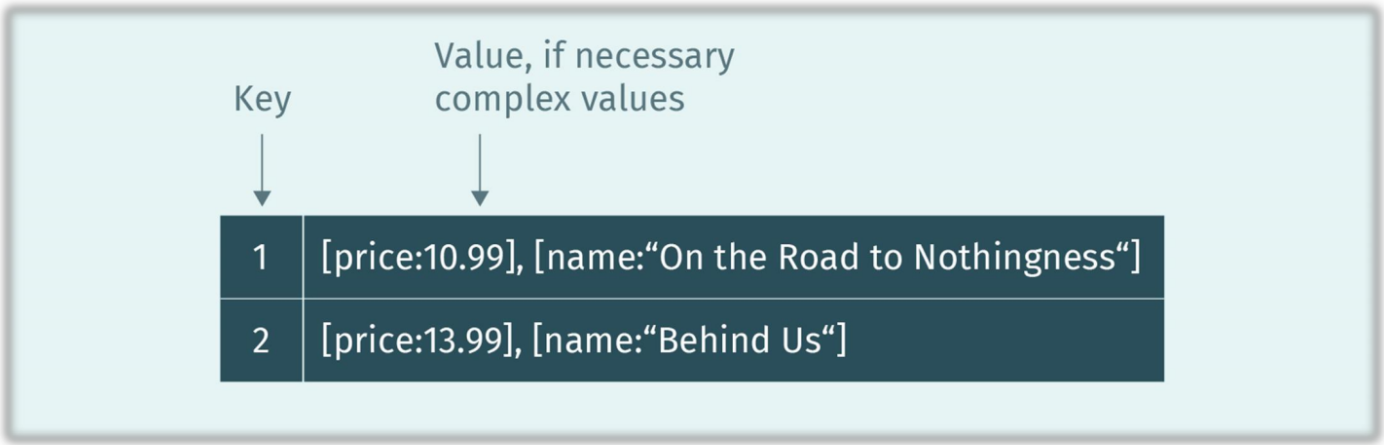
ACID (Relational model)

- Atomicity
- Consistency
- Isolation
- Durability

BASE (Non-relational model)

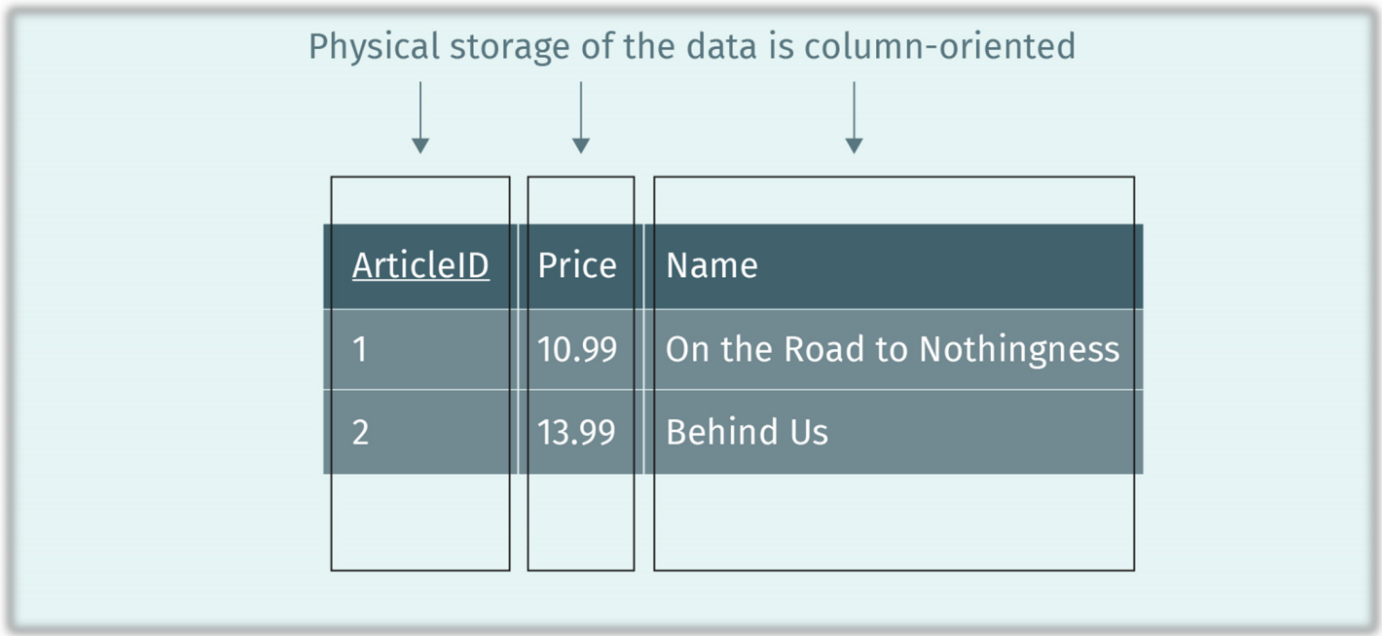
- Basically available
- Soft state
- Eventual consistency

NOSQL SYSTEM TYPES: KEY-VALUE SYSTEM



Source of the graphic: Course Book DLBCSDMD01, p.140

NOSQL SYSTEM TYPES: WIDE COLUMN STORES



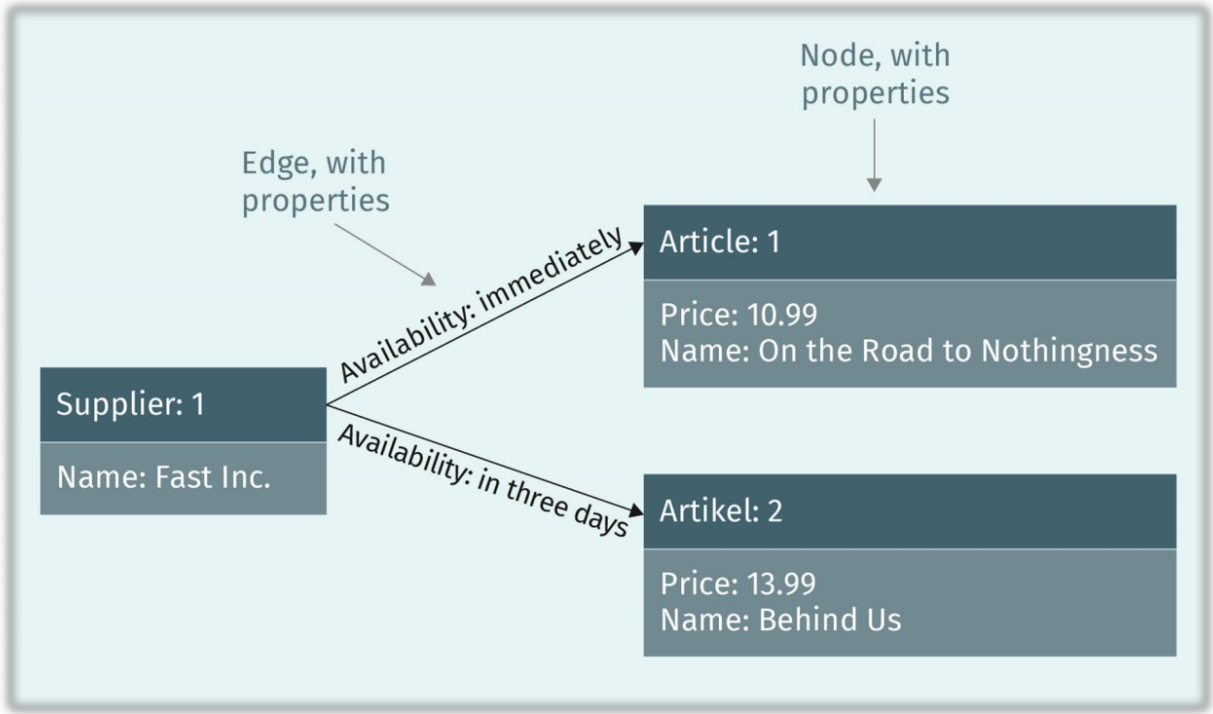
Source of the graphic: Course Book DLBCSDMD01, p.141

NOSQL SYSTEM TYPES: DOCUMENT STORES- JSON

```
{
"ID": "1" {
      "price" : "10.99"
      "name" : “On the Road to Nothingness“
      }
   "ID":"2" {
   "price" : "13.99"
   "name" : “Behind Us“
   }
}
```

Source of the graphic: Course Book DLBCSDMD01, p.142

NOSQL SYSTEM TYPES: GRAPH DATABASES



Source of the graphic: Course Book DLBCSDMD01, p.143

NoSQL Case Study: Real-Time Social Media Analytics Platform

Background

A company aims to develop a platform for real-time analytics of social media data, processing massive streams from millions of users worldwide.

Challenge

The data is highly unstructured and varied, ranging from text posts and images to user interactions and timestamps. The platform must scale dynamically to accommodate unpredictable workloads and provide fast data access for real-time insights.

NoSQL Case Study: Real-Time Social Media Analytics Platform (continued)

Solution with NoSQL

- **Scalability:** A NoSQL database like Cassandra, which excels at write-intensive operations, can handle large data volumes and high throughput.
- **Flexibility:** Document-based NoSQL databases like MongoDB can store various data types without a predefined schema, facilitating the evolution of data structures.
- **Performance:** NoSQL databases provide low-latency data access, essential for real-time analytics and decision-making.

Outcome

The implementation of NoSQL databases allows the platform to efficiently process and analyze data in real-time, leading to actionable insights and enhanced user experience.

Accessing Data in NoSQL for Analytics Platform

Data Access in NoSQL

To extract real-time insights from the social media analytics platform, data can be accessed and queried using the NoSQL database's native language or APIs.

Example: Querying MongoDB for User Posts

```
// MongoDB query to find posts by a specific user
db.posts.find({
  "user_id": "12345",
  "timestamp": {
    $gte: ISODate("2023-01-01T00:00:00.000Z"),
    $lt: ISODate("2023-02-01T00:00:00.000Z")
  }
}).sort({ "timestamp": -1 })
```

Accessing Data in NoSQL for Analytics Platform (continued)

Real-Time Analytics

Such queries enable the platform to perform real-time analytics, like tracking trending topics, analyzing user sentiment, and identifying influential users.

Accessing Data in NoSQL Databases

Data Access Methods

Accessing data in NoSQL databases typically involves using database-specific APIs or query languages designed for non-relational data models.

Examples

- **Document Databases**: Use of JSON-based query languages like MongoDB's query language.
- **Key-Value Stores**: Simple API calls for setting and retrieving data by key, such as Redis commands.
- **Column-Family Stores**: Query languages like CQL for Apache Cassandra to interact with columns and rows.
- **Graph Databases**: Use of graph-specific query languages like Cypher for Neo4j to manage highly connected data.

Accessing Data in NoSQL Databases (continued)

Integration

Many NoSQL databases provide drivers and integrations for popular programming languages, making it easier to access and manipulate data programmatically.

REVIEW STUDY GOALS

- Explain the meaning and the need for NoSQL.
- Discuss the difference between constraints of relational and non-relational data models.
- Explain the differences among different available NoSQL system types.

Meaning and Need for NoSQL

What is NoSQL?

NoSQL, standing for "Not Only SQL," is a category of database management systems that do not require a fixed schema, are optimized for insertions and retrieval, and are designed to scale out by distributing data across many machines.

Why NoSQL?

The need for NoSQL arises from the limitations of traditional RDBMS in handling big data and real-time web applications. NoSQL databases are built to handle a large volume of data that does not fit into traditional database schemas and to support scalable, distributed architectures.

Constraints of Relational vs. Non-Relational Data Models

Relational Model Constraints

Relational models enforce ACID properties, requiring a strict schema, data consistency, and relationships between tables which can limit flexibility and scalability.

Non-Relational Model Constraints

Non-relational models, or NoSQL databases, are more flexible with BASE properties, do not require a fixed schema, and are designed to scale horizontally, which makes them suitable for large-scale data distribution and for applications that require high throughput and flexibility.

Differences Among NoSQL System Types

Document-oriented Databases

Store data as documents, typically in JSON or XML format, allowing for nested data structures. Example: MongoDB.

Key-Value Stores

Store data as a collection of key-value pairs, optimized for retrieval by key. Example: Redis.

Column-oriented Databases

Store data in columns rather than rows, which is efficient for analytics and operations on large datasets. Example: Cassandra.

Graph Databases

Designed to store and navigate relationships, they represent data in graph structures with nodes, edges, and properties. Example: Neo4j.

TRANSFER TASK

Convert the following part of a job offer letter into a JSON document:
"Dear Mr. Johns, We are delighted to offer you the sales manager position at ABC bank, Branch 4 in Berlin, with an annual salary of €122,000 in addition to the health benefits and vacation time. The start date for this position will be on January 23. Your supervisor will be Mme. Franklin and her office number is 215."

Job Offer in JSON Format

```
{
  "recipient": "Mr. Johns",
  "message": {
    "greeting": "We are delighted to offer
       you",
    "position": "sales manager",
    "company": "ABC bank",
    "location": {
      "branch": "Branch 4",
      "city": "Berlin"
    },
    "compensation": {
      "salary": {
        "annual": "   122,000"
      },
  }
```

Job Offer in JSON Format (continued)

```
{
        "benefits": ["health benefits", "
           vacation time"]
    },
    "start_date": "January 23",
    "supervisor": {
      "name": "Mme. Franklin",
      "office_number": "215"
    }
  }
}
```