Introduction to NoSQL Databases

Models, Trade-offs, and Polyglot Persistence

Prof. Ing. Lelio Campanile, PhD

Corso di Laurea in Data Analytics
Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli

A.A. 2025/2026

Lecture Goals and Context

Lecture Goals

By the end of this lecture, you will be able to:

- Explain *why* NoSQL systems emerged historically
- Distinguish the four main NoSQL families
- Compare SQL and NoSQL conceptually
- Discuss **CAP** and **PACELC** trade-offs
- Recognize when *not* to use NoSQL

Università degli Studi della Campania *Luigi Vanvitelli*

Where We Are in the Course

So far we have studied:

- The **relational model**
- **Conceptual design** – Entity-Relationship modeling
- **Logical design** – normalization
- **SQL** – DDL, DML, joins, transactions, ACID

A New Question

What happens when the relational model is no longer the best tool for the job?

This is the question NoSQL tries to answer.

Università degli Studi della Campania *Luigi Vanvitelli*

A Brief History of NoSQL

A Brief Historical Detour

The term **NoSQL** is not as old as you might think.

Timeline

- **2006** – Google BigTable paper
- **2007** – Amazon Dynamo paper
- **2009** – San Francisco meetup coins *"NoSQL"*
- **2010s** – Explosion of non-relational systems
- **2020s** – Convergence: NoSQL adopts ACID

Università degli Studi della Campania *Luigi Vanvitelli*

NoSQL Was Not a Revolution

Key Principle

NoSQL was **not** an ideological revolt against SQL.

It was a **pragmatic response** to scale problems that relational systems struggled to solve.

Why Relational Was Challenged

Web-scale applications in the late 2000s faced new pressures:

- **Volume** – petabytes, not gigabytes
- **Velocity** – millions of writes per second
- **Variety** – semi-structured data, JSON
- **Distribution** – global users, global data
- **Evolution** – schemas changing every week

Where Relational Still Excels

The relational model is **still the right choice** for most enterprise systems.

Typical Domains

- Banking and accounting
- ERP and CRM
- University and government records
- Inventory and order management

Any domain where **transactional integrity** is non-negotiable.

Where Relational Struggles

Some scenarios push the relational model to its limits:

- **Social networks** – deep graph traversals
- **IoT and sensors** – high-velocity time series
- **Content platforms** – heterogeneous documents
- **Recommendation engines** – complex similarity
- **Real-time analytics** – sub-second latency

Università degli Studi della Campania *Luigi Vanvitelli*

What is NoSQL?

What is NoSQL?

The acronym is widely interpreted as:

Definition

"Not Only SQL"

- NoSQL is **not a single technology**.
- It is a **family** of database systems.

Common Traits of NoSQL Systems

Shared characteristics

- Non-relational data models
- Flexible or schema-less storage
- Designed for horizontal scalability
- Often distributed by default
- Trade-offs on classical ACID guarantees

Flexible Schema

Relational vs NoSQL

- In a **relational** database, the schema is *declared upfront* and enforced on every row.
- In most **NoSQL** systems, each record can have a *different structure*.

Flexible Schema – Example

```
{ "id": 1, "name": "Alice", "email": "alice@example.com" }
```

```
{ "id": 2, "name": "Bob",   "phone": "+39 333 1234567" }
```

```
{ "id": 3, "name": "Carol", "tags": ["vip", "newsletter"] }
```

Consequence

This shifts the burden of consistency to the **application layer**.

Horizontal Scalability

Two ways to grow a database:

Vertical scaling

Bigger CPU, more RAM.

Easy, but expensive and bounded.

Horizontal scaling

Add more machines.

Cheaper at scale, but more complex.

Insight

Most NoSQL systems are designed for **horizontal scaling from day one**.

Distribution and Availability

Once data is **distributed** across many nodes:

- The system tolerates **hardware failures**
- It can serve requests **closer to users**
- It can handle **higher throughput**

But distribution introduces new challenges.

New Challenges of Distribution

- Network partitions
- Replica synchronization
- Trade-offs between consistency and availability

This is where the **CAP theorem** enters the picture.

Distribution Trade-offs

CAP and PACELC

The CAP Theorem

Origin

Formulated by **Eric Brewer** (2000). Proved by Gilbert and Lynch (2002).

Three desirable properties:

- **C** – every read sees the most recent write
- **A** – every request gets a response
- **P** – the system works despite network failures

The CAP Trade-off

In a distributed system, **partitions will happen**.

So the real choice is:

The Real Question

*When the network splits, do we prefer **consistency** or **availability**?*

CAP – A Common Misconception

You will often hear:

Popular Phrasing

"Pick two out of three."

This is misleading.

Correction

Partition tolerance is not optional – the network *will* fail.

Università degli Studi della Campania *Luigi Vanvitelli*

CAP – The Correct Reading

The trade-off is **binary and conditional**:

- **During a partition** $\rightarrow$ choose C or A
- **Without a partition** $\rightarrow$ you can have both

Reference

Brewer clarified this in *"CAP Twelve Years Later"* (2012).

Università degli Studi della Campania *Luigi Vanvitelli*

Beyond CAP – PACELC

Daniel Abadi (2012) extended CAP with a more realistic model:

PACELC

- **If Partition** → choose between **A** and **C**
- **Else** → choose between **L**atency and **C**onsistency

PACELC – Real Systems

System	Classification	Priority
Cassandra	PA / EL	Availability + low latency
MongoDB	PA / EC	Availability + consistency at rest
HBase	PC / EC	Consistency always

The Four NoSQL Families

The Four NoSQL Families

Taxonomy

1. **Key-Value stores**
2. **Document databases**
3. **Wide-Column stores**
4. **Graph databases**

Each family targets a different class of problems.

Family 1 – Key-Value Stores

The simplest NoSQL model:

```
key  ->  value
```

- The value is **opaque** to the database.
- Operations are minimal: `GET`, `PUT`, `DELETE`.

Università degli Studi della Campania *Luigi Vanvitelli*

Key-Value – Strengths and Limits

Strengths

- Extreme speed
- Simple horizontal scaling

Limitations

- No querying *inside* values
- No joins, no rich indexing

Università degli Studi della Campania *Luigi Vanvitelli*

Key-Value – Typical Uses

Use cases

- **Session storage** – user_id → session
- **Caching** – query_hash → result set
- **Feature flags** – flag → boolean
- **Leaderboards** – game_id → scores
- **Rate limiting** – ip → count

Technologies

Redis, DynamoDB, Riak, Memcached.

Università degli Studi della Campania *Luigi Vanvitelli*

Key-Value – Example

```
SET user:1001 "{\"name\":\"Alice\",\"age\":21}"
```

```
GET user:1001
```

Returns:

```
{ "name": "Alice", "age": 21 }
```

Note

The database does **not** understand the JSON structure.

Family 2 – Document Databases

Definition

Data is stored as **self-contained documents**, typically in JSON or BSON.

Each document is **identified by a key** *and* **introspectable** by the database.

Document – Example

```
{
  "student_id": 1001,
  "name": "Alice",
  "enrollment_year": 2024,
  "exams": [
    { "course": "Databases",        "grade": 28 },
    { "course": "Machine Learning", "grade": 30 }
  ]
}
```

Insight

You can query, index, and aggregate **inside** the document.

Document vs Relational

The same information in a relational schema:

```
STUDENTS   (student_id, name, enrollment_year)
COURSES    (course_id, course_name)
EXAMS      (student_id, course_id, grade)
```

- A transcript query needs **two joins**.
- In a document model: **one read, no joins**.

Document – Embed vs Reference

The central design decision:

Embed

Nest related data inside the parent.

Reference

Store a pointer (id) to another document.

Embed vs Reference – Guidelines

Heuristics

- **1-to-few** → embed
- **1-to-many, rarely updated** → embed
- **1-to-many, frequently updated** → reference
- **Many-to-many** → reference

This mirrors the *denormalization* decisions of relational design.

Università degli Studi della Campania *Luigi Vanvitelli*

Document – Strengths and Limitations

Strengths

- Natural fit for web apps
- One read = full aggregate
- Schema evolves without migrations
- Excellent for content-heavy apps

Limitations

- Data duplication if embedded aggressively
- Update anomalies across documents
- Cross-document joins are limited

Technologies

MongoDB, CouchDB, DocumentDB, Firestore.

Family 3 – Wide-Column Stores

A common point of confusion – let us be precise:

Three different "column" concepts

- **Row-oriented** – relational DBs (PostgreSQL, MySQL)
- **Column-oriented (analytical)** – Parquet, ClickHouse
- **Wide-column (NoSQL)** – Cassandra, HBase

Warning

Wide-column is **not** the same as column-oriented analytics.

Università degli Studi della Campania *Luigi Vanvitelli*

Wide-Column – Model

A wide-column store is a **sparse, distributed, multi-dimensional map**:

```
(row_key, column_family, column, timestamp) -> value
```

Rows are **sparse** – each row can have completely different columns.

Wide-Column – Use Cases

When to use

- Time-series data
- IoT telemetry
- Write-heavy logs
- Large-scale activity feeds

Technologies

Apache Cassandra, Apache HBase, ScyllaDB, Google BigTable.

Family 4 – Graph Databases

When the **relationships** between entities are as important as the entities themselves.

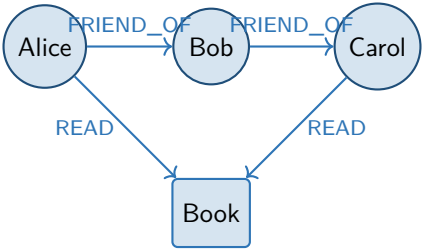
Two primitives

- **Nodes** – entities
- **Edges** – typed, directed relationships

Both can carry **properties**.

Università degli Studi della Campania *Luigi Vanvitelli*

Graph – Visual Example



Edges typed as `FRIEND_OF` and `READ`.

Graph – Why It Matters

Insight

Queries that would require **many recursive joins** in SQL become **natural traversals** in a graph database.

"Friends of friends who read the same book as Alice"
$\rightarrow$ *a few lines of Cypher.*

Graph – Typical Uses

Use cases

- **Social networks** – friends, communities
- **Recommendations** – collaborative filtering
- **Fraud detection** – suspicious patterns
- **Knowledge graphs** – semantic entities
- **Infrastructure mapping** – dependencies

Technologies

Neo4j, Amazon Neptune, ArangoDB, TigerGraph.

Università degli Studi della Campania *Luigi Vanvitelli*

Query Languages

Query Languages – A Quick Tour

Each family has its own query language:

Mapping

- **Relational** → SQL (ISO standard)
- **Document** → MQL, JSONPath
- **Wide-Column** → CQL (SQL-like syntax)
- **Graph** → Cypher, GQL

A Note on CQL and GQL

Beware

CQL *looks* like SQL but does **not** support real joins.
Same syntax, different semantics.

Maturity Signal

GQL (ISO/IEC 39075:2024) is the first ISO standard for graph queries.
A sign of the field's maturity.

Università degli Studi della Campania *Luigi Vanvitelli*

SQL vs NoSQL

SQL vs NoSQL – Schema and Model

Aspect	**SQL**	**NoSQL**
Schema	Fixed, enforced	Flexible
Data model	Tables	Multiple models
Joins	First-class	Limited or absent

SQL vs NoSQL – Scale and Guarantees

Aspect	**SQL**	**NoSQL**
Scaling	Mostly vertical	Mostly horizontal
Transactions	ACID (strong)	Historically BASE
Consistency	Strong default	Often tunable

Università degli Studi della Campania *Luigi Vanvitelli*

SQL vs NoSQL – Best Use

SQL is best for

Structured, transactional workloads.

NoSQL is best for

Large-scale, flexible, distributed workloads.

ACID vs BASE

ACID – A Reminder

The classical guarantees of relational transactions:

ACID

- **A – Atomicity** – all-or-nothing
- **C – Consistency** – valid $\rightarrow$ valid state
- **I – Isolation** – no interference
- **D – Durability** – survives failures

Università degli Studi della Campania *Luigi Vanvitelli*

ACID – A Limitation

Key Point

ACID was designed for **single-node, transactional** workloads.

It does not scale trivially across a distributed cluster.

This led to a weaker, more scalable model.

BASE – The NoSQL Alternative

Early NoSQL systems adopted a relaxed model:

BASE

- **B – Basically Available**
- **S – Soft state**
- **E – Eventual consistency**

BASE – In Practice

Everyday Example

When you “like” a post on a social network, your friends may see the new count **after a few seconds**.

That delay is **eventual consistency** in action.

For many web-scale workloads, this trade-off is acceptable.

ACID vs BASE – The Modern Picture

The strict dichotomy is **outdated**.

Recent evolution

- **MongoDB** – multi-document ACID since 4.0 (2018)
- **Cassandra** – lightweight transactions via Paxos
- **CockroachDB**, **YugabyteDB** – "NewSQL"

Università degli Studi della Campania *Luigi Vanvitelli*

The Real Modern Question

Old Question

"ACID or BASE?"

Better Question

Which consistency guarantees do I need, on which operations, at which cost?

Università degli Studi della Campania *Luigi Vanvitelli*

A Worked Example

The Library System Revisited

A Worked Example – Library

Recall the **library system** from earlier in the course.

Relational design

```
STUDENTS (student_id, name, email)
BOOKS    (book_id, title, author, isbn)
LOANS    (loan_id, student_id, book_id, loan_date)
```

Retrieving active loans requires a **join**.

Library System – Document Version

```
{
  "student_id": 1001,
  "name": "Alice",
  "active_loans": [
    { "book": "Database Systems",
      "isbn": "978-0136086208",
      "loan_date": "2026-05-14" }
  ]
}
```

Insight

One document, one read, the full active state.

Università degli Studi della Campania *Luigi Vanvitelli*

Library – Document Wins On

Advantages

- Read performance for the most frequent query
- Natural mapping to the application object
- Schema flexibility per student

Library – Document Loses On

Trade-offs

- *"How many copies of book X are on loan?"* $\rightarrow$ must scan all students
- Updating book metadata in one place – duplicated
- Strict referential integrity – no foreign keys

Library – The Lesson

The Lesson

The right model depends on the **dominant access pattern**.

Not on which technology is “newer” or “trendier”.

When Not to Use NoSQL

and Polyglot Persistence

When *Not* to Use NoSQL

Choose a **relational database** when:

- Data is **highly structured** and stable
- **Multi-table transactions** are central
- **Referential integrity** must be enforced
- **Ad-hoc joins** are frequent
- The dataset fits on one large machine

Università degli Studi della Campania *Luigi Vanvitelli*

A Useful Sanity Check

Rule of Thumb

"You probably don't have big data."

PostgreSQL on a well-provisioned server handles **far more** than most projects ever need.

Start simple. Scale when forced to.

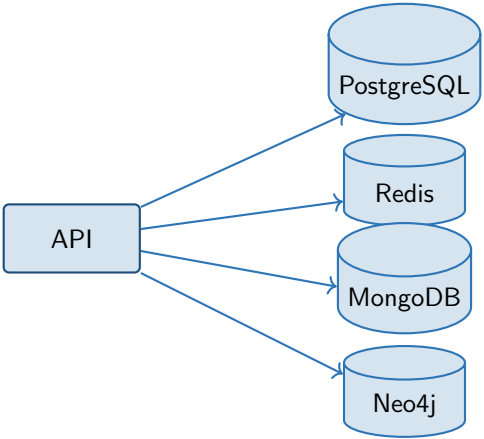
Polyglot Persistence

Definition

Using different databases for different parts of the same application – each chosen for its strengths.

Modern systems rarely commit to a **single** database technology.

Polyglot Persistence – Architecture



Università degli Studi della Campania *Luigi Vanvitelli*

Polyglot – E-commerce Example

Each system does what it does best

- **PostgreSQL** – orders, payments → ACID
- **Redis** – sessions, cache → low latency
- **MongoDB** – product catalog → flexible
- **Neo4j** – recommendations → graph
- **Elasticsearch** – full-text search

Università degli Studi della Campania *Luigi Vanvitelli*

Polyglot – The Cost

Operational Complexity

- More services to deploy
- More systems to monitor
- More integrations to keep in sync

Adopt polyglot only when **justified by evidence**.

Choosing a Database

Choosing a Database – Step 1

First Question

What is the dominant access pattern?

Mapping

- Point lookups → **key-value**
- Aggregate reads by id → **document**
- Heavy writes, time-series → **wide-column**
- Relationship traversal → **graph**
- Joins and ad-hoc queries → **relational**

Università degli Studi della Campania *Luigi Vanvitelli*

Choosing a Database – Step 2

Then ask

- What consistency do I need, on which operations?
- What is my realistic data scale today and in 2 years?
- What operational cost can I afford?

Key Takeaways

Key Takeaways – 1

What NoSQL Is

- NoSQL is a **family** of systems, not one technology
- It emerged as a **pragmatic response** to web-scale
- Four main models: **key-value, document, wide-column, graph**

Key Takeaways – 2

Trade-offs and Design

- **CAP** is a useful intuition; **PACELC** is more realistic
- The **ACID vs BASE** dichotomy has softened
- Relational databases are **still the right default**
- Real architectures often combine technologies – **polyglot persistence**

References

Suggested Readings – Foundational

Books

- Sadalage & Fowler – *NoSQL Distilled*, Addison-Wesley, 2012
- Kleppmann – *Designing Data-Intensive Applications*, O'Reilly, 2017

Università degli Studi della Campania *Luigi Vanvitelli*

Suggested Readings – Original Papers

Seminal papers

- Brewer – *"CAP Twelve Years Later"*, IEEE Computer (2012)
- Abadi – *"Consistency Tradeoffs..."*, IEEE Computer (2012)
- DeCandia et al. – *"Dynamo"*, SOSP (2007)
- Chang et al. – *"Bigtable"*, OSDI (2006)

Università degli Studi della Campania *Luigi Vanvitelli*

Prof. Ing. Lelio Campanile
Assistant Professor (RTDa)

Dipartimento di Matematica e Fisica
Università degli Studi della Campania Luigi Vanvitelli

Databases and Information Systems
Corso di Laurea in Data Analytics

Questions?

lelio.campanile@unicampania.it

Next lecture: course wrap-up and exam preparation.

Use of the present material requires authorization from the author. No modifications are allowed, especially to author identification. The author assumes no responsibility for the use of this material.