in28minutes

Getting Started With Data and Intelligence

What is Data?

Data: Raw facts, figures, and information

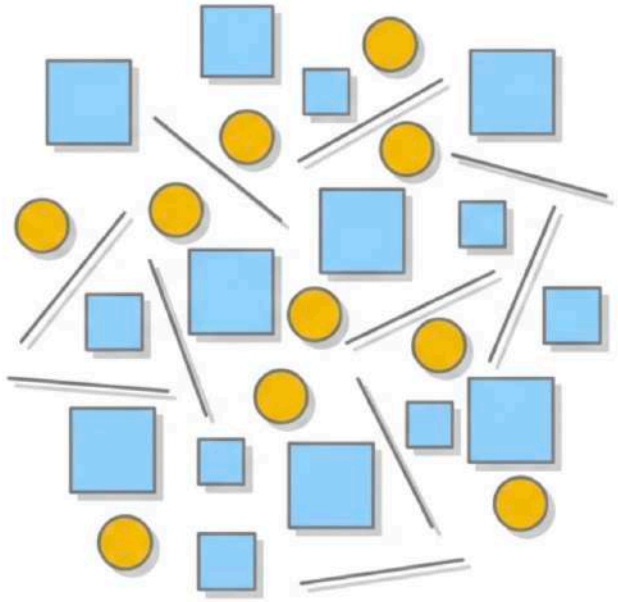
- **Examples**: Numbers, text, images, ..

Volume: We are creating massive amounts every second

- **Growth**: Exponential increase in data generation

Variety: Comes in many forms

- **Structured**: Tables, SQL databases
- **Semi-structured**: No rigid structure (JSON, XML)
- **Unstructured**: Photos, videos, archives



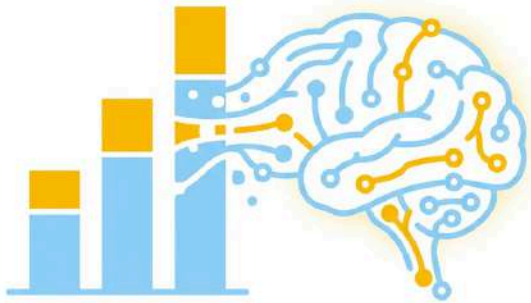
What is Intelligence?

Intelligence: Actionable Insights derived from data

- **Goal**: Make better decisions

Data vs Intelligence:

- **Data**: What you have - the raw numbers
  - *"It is 30°C (86°F) outside"*
- **Intelligence**: What happened and What can you do
  - *I should eat an ice cream*
- **Example**:
  - **Data**: "Customer X cancelled subscription"
  - **Intelligence**: "Customer X cancelled because of price; offer a discount to retain them"



How can we get Intelligence From Data?

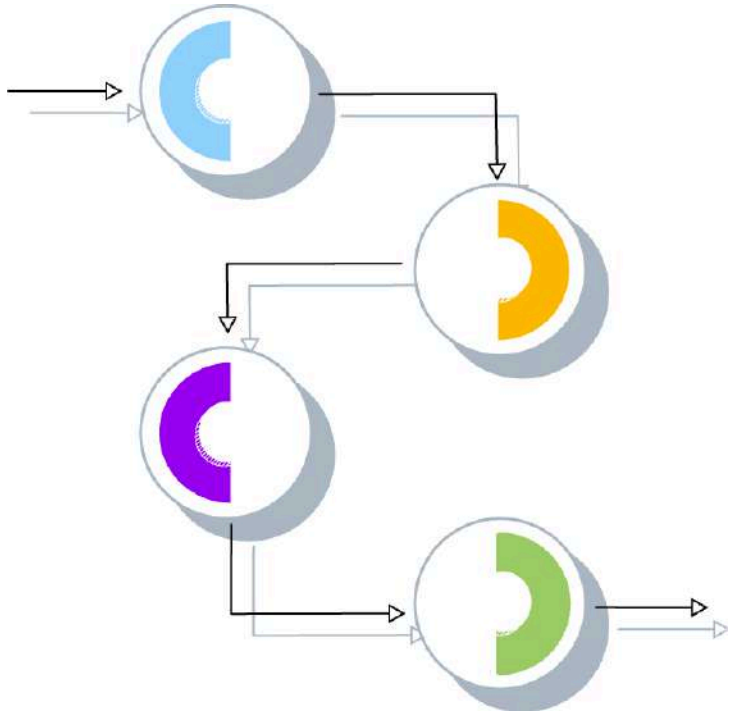
Answer: Analytics & AI

Analytics: Understanding the past

- **Descriptive**: What happened? (*"Engine is overheating."*)
- **Diagnostic**: Why did it happen? (*"Coolant tank is empty."*)

AI/Machine Learning: Predicting the future

- **Predictive**: What will happen? (*"If you drive 10 more miles, the engine will fail."*)
- **Prescriptive**: How can we make it happen? (*"Stop now and refill the coolant."*)



Ready To Explore Data?

Journey: We are starting a deep dive into the world of Data

Key Topics:

- **Data Reliability**: Understand Availability, Durability, Scalability and Consistency
- **Formats**: What are the different data formats?
- **Storage**: Where to store data? (Databases, Data Warehouses, ..)
- **Intelligence**: How can you get intelligence from data?

Let's Go!

in28minutes

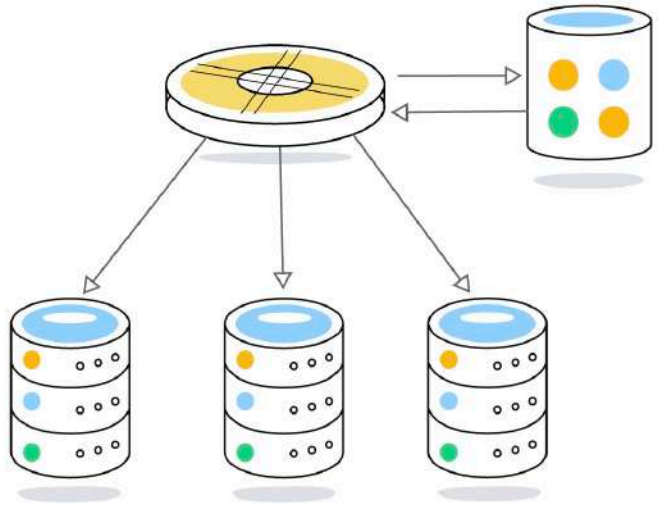
Getting Started with Data Fundamentals

We Do Not Want To Lose Data

Goal: Build Reliable Data Systems

Key Pillars: Four Key Pillars

- **Availability**: Can users access data store when they need it?
- **Durability**: Will the data still exist years from now?
- **Scalability**: Can we handle more data and users without slowing down?
- **Consistency**: Will all users see the same, correct version of the data?



Let's Get on a Journey: Understand the four key pillars

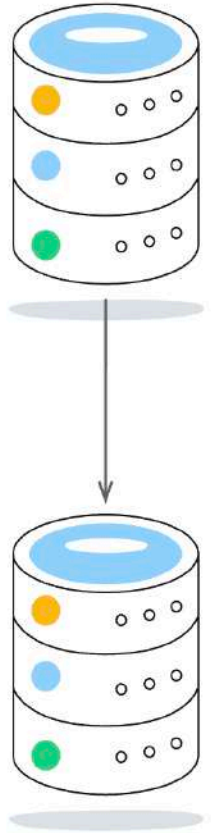
What is Availability?

Availability: Will your data be accessible when your application needs it

Measured As: Uptime percentage over a specified period

Typical Targets:

- **99.95%** – About 22 minutes of downtime per month
- **99.99% (Four Nines)** – About 4.5 minutes of downtime per month
- **99.999% (Five Nines)** – About 26 seconds of downtime per month



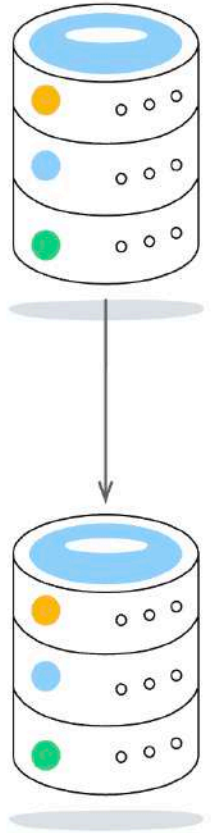
Improving Availability – The Challenge of Nines

Typical Target: Most Systems Target 99.99% (Four Nines) availability

- **99.999% is tough**: Requires reducing monthly downtime from 4.5 minutes(99.99%) to 26 seconds (99.999%)

How to Improve: Choices include

- **Redundancy**: Standby instances ready to use
- **Geo Distribution**: Deploy services across multiple zones or regions
- **Automatic Failover**: Implement health checks and automated switch-over logic



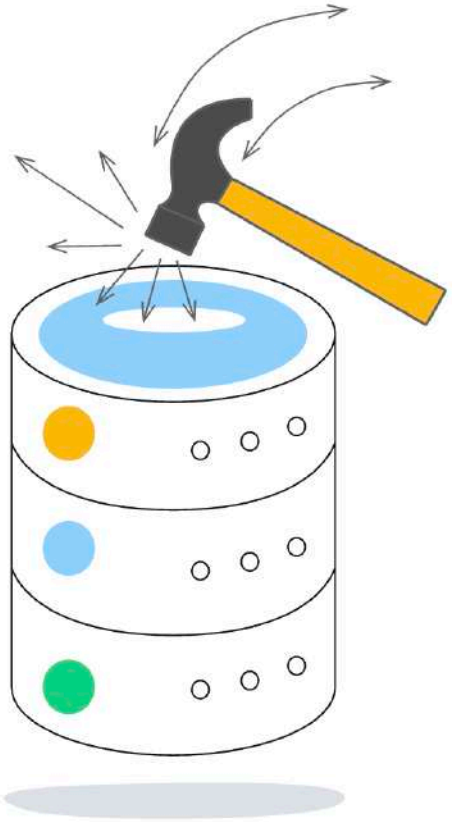
What is Durability?

Durability: Will your data still exist years from now, even through failures or disasters

- **Measured As**: The probability of losing data over a specific period
- **Typical Target**: **99.999999999% (Eleven Nines)** (If you store one million files for ten million years, you expect to lose only one file)

Why It Matters:

- **Irreversible**: Once data is lost, it is generally unrecoverable
- **Compliance**: Financial records, medical data, legal archives, and compliance requirements



How To Improve Durability

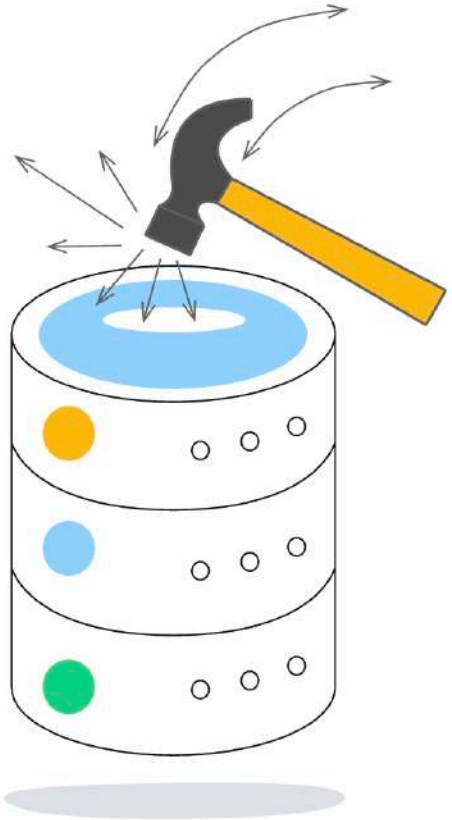
Redundancy: Store multiple copies of data

Geo Distribution: Distribute copies across separate zones and regions

- **Impact**: Protects against a single data center failing or a regional disaster

Versioning: Keep previous states of data

- **Protection**: Recover quickly from accidental deletes



Availability vs Durability – The Bank Analogy

The Scenario: You go to ATM

Availability: Is the ATM working right now?

Durability: Is your account balance record safe?

- **Failure**: The bank loses your record entirely - your money "disappears" (Permanent)

Key Lesson: You can survive an app/data being "unavailable" for an hour, but you cannot survive data being "indurable"

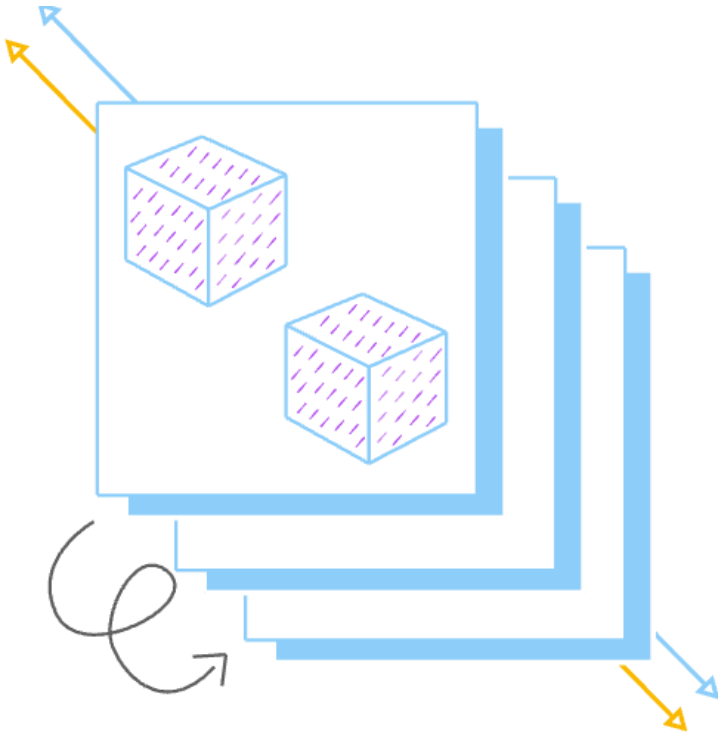
Availability vs Durability

	Availability	Durability
Focus	Data is **accessible now**	Data is **never lost**
Typical Goal	99.99% (4 nines)	99.999999999% (11 nines)
Goal	Avoid downtime	Prevent permanent data loss
Failure Impact	Temporary service outage	Permanent data loss
Achieved Through	Redundancy, Geo Distribution, Failover	Redundancy, Geo Distribution, Versioning
Applicable To	Apps, Data	Data

Scalability for Data Stores – Overview

Scalability: Handle more without slowing down

- More data
- More users
- More connections
- More queries



Two Approaches

- **Vertical scaling** – Bigger server
- **Horizontal scaling** – More servers

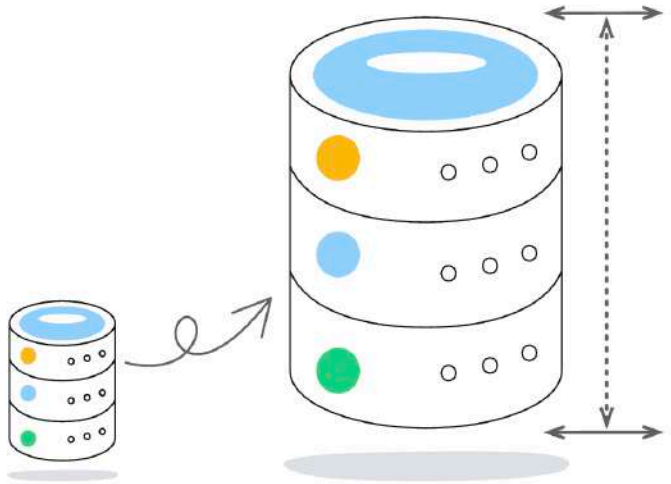
What is Vertical Scaling?

Concept: Increase the power of a single database server instance

- **Action**: Upgrade CPU, RAM, and Disk capacity

Simplicity: Easiest way to scale - typically requires zero application code changes

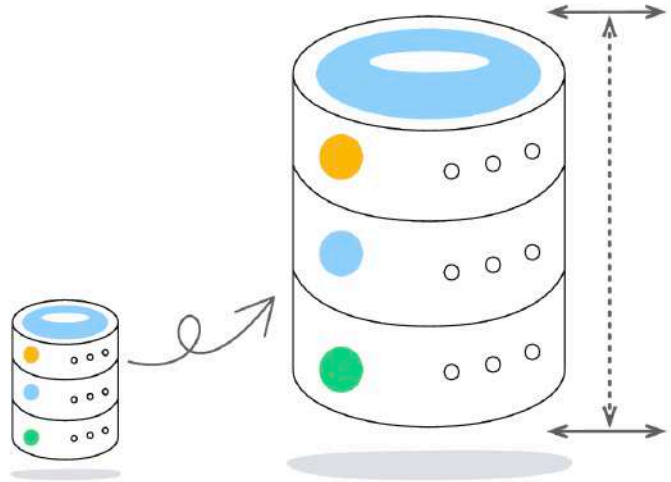
Temporary Downtime: Most often requires temporary downtime for the instance to apply the changes



Vertical Scaling – Limits and Use Cases

Limitations:

- **Hardware Ceiling**: You will eventually hit the maximum physical limits of a single server
- **Single Point of Failure**: If that one server fails, the entire database is inaccessible
- **Cost**: The biggest machines are disproportionately expensive



Best For:

- **Smaller Workloads**: Databases with predictable, limited growth
- **Simple Architecture**: When you need simplicity and don't want complexity

What is Horizontal Scaling?

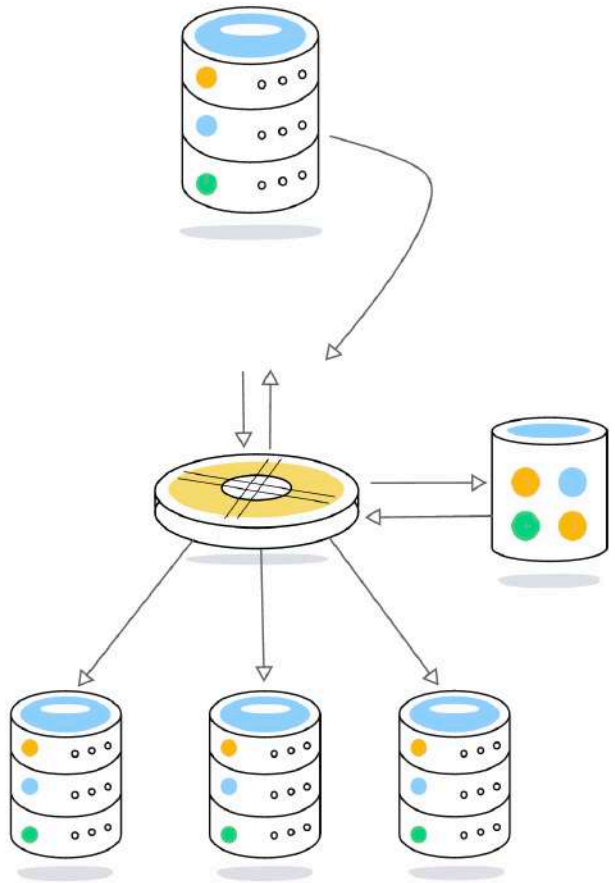
Idea: Add more servers and distribute data/traffic

Pros

- **High throughput** – Parallel reads/writes
- **Resilience** – No single bottleneck

Cons

- **Complexity** – Multiple servers to manage
- **Consistency** – How do ensure that data across multiple servers is the same?



Horizontal Scaling Example - 1 – Read Replica

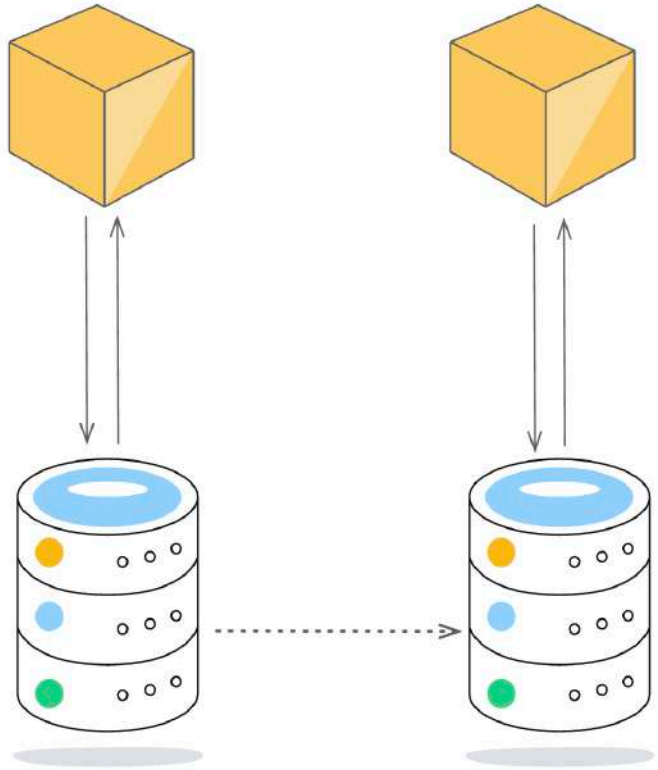
Concept: Create read-only copies of the primary database

- **Primary server**: Handles all **Writes**
- **Read Replicas**: Handle all **Reads** (SELECT queries)

Benefit: Primary can focus solely on writes, providing better performance

Limitation: Doesn't solve write bottlenecks or overall database size limitations

Use Cases: Heavy reporting, analytics



Horizontal Scaling Example - 2 – Distributed Data Stores

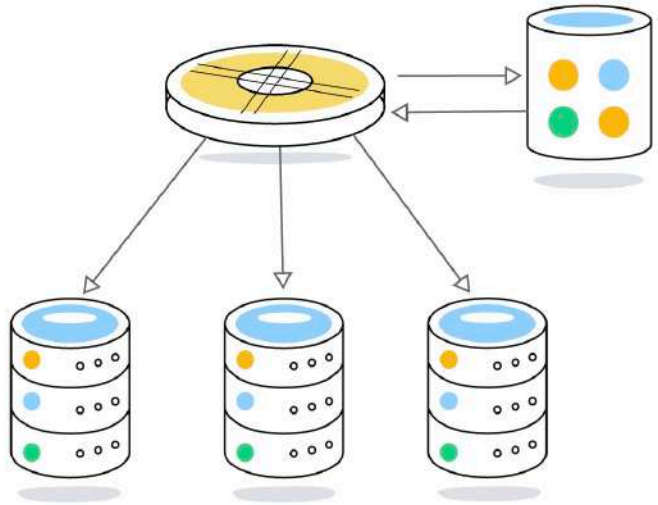
Distributed Database: A database that distributes data across **many servers**

- **Key Feature**: Automatically splits data (and queries) across all servers

Benefit:

- **Unlimited Capacity**: Easily handles massive data size and high transaction volume
- **High availability**: Replicated across zones/regions

Question: How do we ensure that all servers have same data while providing good enough performance?



Setting Stage for Consistency

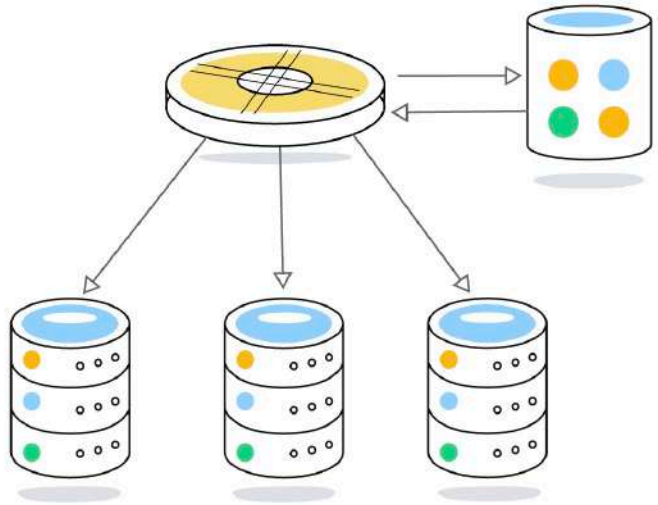
Goal: Make data highly available and durable

Easiest Way – Redundancy

- Add more copies of your data across zones or regions

BUT… Is It Free?

- Every update needs to be **copied to replicas**
- What if replicas are in different regions?
  - Updates take time (Network delays come into play)

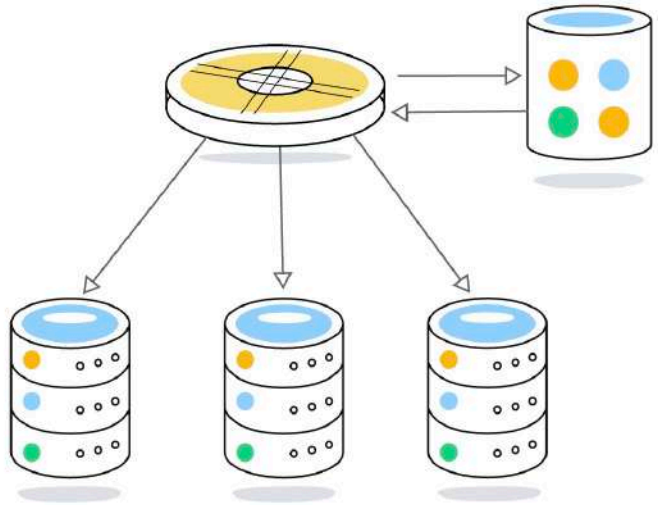


Different Systems Have Different Consistency Needs

Scenario	Consistency Need	Explanation
Bank Account Balance	Strong Consistency	Data must be correct and consistent instantly across all users and systems
Inventory Management	Strong Consistency	Critical for knowing the exact number of items available for sale
Social Media Feed	Eventual Consistency	Small delay in seeing a new post or comment is acceptable
Website Personalization	Eventual Consistency	User profile updates can take a few seconds to reflect on all servers

What is Strong Consistency - Getting Started

Strong Consistency: All replicas return the same, most recent value after a write



Trade-offs

- **NO Stale Reads**: Every read reflects the latest data
- **Performance overhead** - Slower writes due to coordination between replicas
- **Best for** - Systems prioritizing correctness over speed

Typical Use Cases:

- **Banking systems** – Account balance updates
- **Financial trading** – Real-time transaction accuracy

What is Eventual Consistency?

Eventual Consistency: All replicas will eventually reflect the latest value

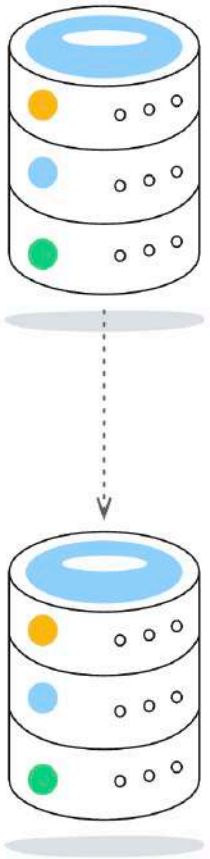
Trade-offs

- **High Performance** – Faster writes, reduced latency
- **Temporary Inconsistency** – Data slightly outdated

Best for - Systems prioritizing speed over immediate accuracy

Typical Use Cases

- **Social Media Feeds** – Likes/comments with delay
- **E-Com Reviews** – Product ratings sync gradually



Exploring Data Stores - Check Your Understanding

Scenario	Solution
What is good Availability?	99.99% (4 9's) - 4 and 1/2 minutes of downtime in a month 99.999% (5 9's) - 26 seconds of downtime in a month
What is good Durability?	Typically, 11 9's durability (99.999999999%) is expected. Once we lose data, it is gone.
What are we measuring here: "Do you get the most recent, updated data irrespective of the copy you are querying against?"	Consistency
What is Eventual Consistency?	A little lag - few seconds - before updates are available in all replicas. In the intermediate period, different replicas might return different values.

Data Reliability in the Cloud - Scenarios

Scenario	What is being discussed?
I can't access the website right now	Availability Issue
My uploaded photo disappeared permanently	Durability Issue
I updated my profile, but my friend still sees the old one	Eventual Consistency
I need to handle 10x more users next week	Scalability
I need the exact stock price immediately after update	Need for Strong Consistency
Strong or Eventual Consistency: Financial Trading Platform	Strong Consistency
Strong or Eventual Consistency: Social media feed (Likes count)	Eventual Consistency (Speed > Immediate Precision)

Wrapping Up – Building Reliable Data Systems

Availability → Access to data anytime

Durability → Data never lost

Scalability → Data store keeps pace with with increase in data size and numbers of users

Consistency → Same truth everywhere

Trade-off → Depending on application needs!