Introduzione

1. I sistemi : definizioni ed esempi

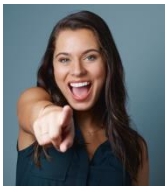
In informatica per sistema si intende, **un insieme di componenti interconnessi che lavorano insieme per raggiungere un determinato obiettivo.** E' una cosa reale che esiste e funziona.
Spesso si presenta come unico oggetto od entità, persona animale o concetto astratto

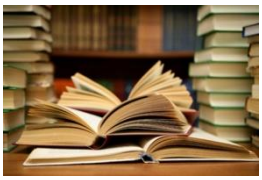
Esempi in informatica:

- Un personal computer, un cellulare, una stampante, un tablet, un server, una rete di computer, un sistema di gestione database, un sistema di elaborazione di transizione online, ecc..
- Una automobile, un robot, uno scooter, una lavatrice, una nave,un trattore, un distributore di bevande o di carburante , ecc..
- Una persona, un animale
- Un edificio, una scuola, una biblioteca, sistema impiantistico (idraulico, elettrico) ecc..









Riassumendo:

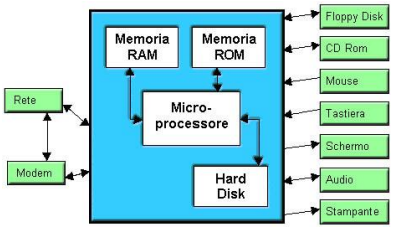
1. Un sistema è composto da diverse parti che hanno funzioni specifiche
2. Le parti sono interconnesse tra loro ed interagiscono per raggiungere l'obiettivo del sistema: ad esempio un'auto è composta da: motore, telaio, organi di trasmissione, carrozzeria, ruote, ecc..
3. Un sistema svolge nel suo complesso una funzione specifica: ad esempio l'auto trasporta le persone, il camion persone e merci, la stampante trasferisce le informazioni su carta

2. I modelli

Un modello è una rappresentazione semplificata di un sistema :
Esempi di modelli possono essere:

- Un diagramma, una tabella, un grafico, un grafo, una piantina di un edificio
- Un insieme di equazioni matematiche
- Un programma (software) con simulazione al computer
- Un disegno al cad, una simulazione al cad
- Un database, un modello OOP (classi ed oggetti)

L'obiettivo di un modello è di aiutarci a capire come funziona un sistema, a progettarlo, o a prevederne il comportamento



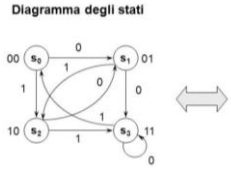
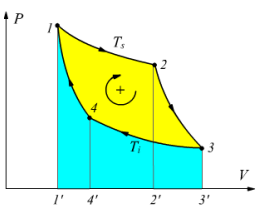
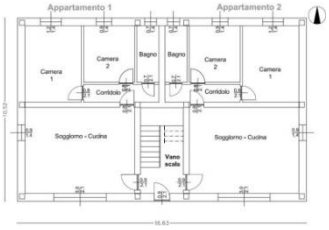


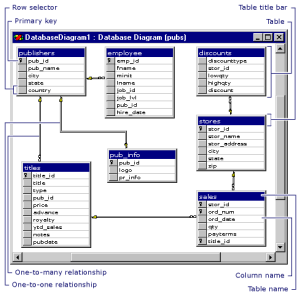
Tabella degli stati

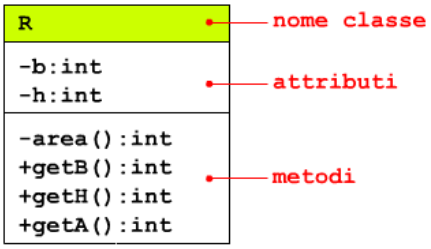
	0	1	U
S_0	S_1	S_2	00
S_1	S_3	S_2	01
S_2	S_1	S_3	10
S_3	S_3	S_0	11





$$\begin{cases} y = 2x + 1 \\ 3x + y = 0 \end{cases}$$





Definizione di database (archivio elettronico)

Un database è un insieme di file in binario salvato su disco"?

No, la definizione di un database come "un insieme di file in binario salvato su disco" è una **descrizione molto semplificata e incompleta**. Sebbene i dati di un database alla fine vengano memorizzati su un supporto fisico come un disco rigido in formato binario, un database è **molto di più** di una semplice collezione di file.
Ecco una definizione più completa e precisa di cosa sia un database:
Un database è una collezione organizzata di dati strutturati, tipicamente memorizzata elettronicamente in un sistema informatico, in modo da poter essere facilmente accessibile, gestita e aggiornata.
Vediamo i punti chiave di questa definizione:

- **Collezione organizzata:** I dati in un database non sono semplicemente ammassati. Sono strutturati in modo logico per rappresentare entità, relazioni e informazioni del mondo reale. Questa organizzazione permette di trovare e gestire i dati in modo efficiente.
- **Dati strutturati:** I dati hanno un formato definito (ad esempio, tabelle con righe e colonne, documenti con campi specifici, grafi con nodi e archi). Questa struttura consente di interrogare e analizzare i dati in modo significativo.
- **Tipicamente memorizzata elettronicamente:** La maggior parte dei database moderni sono implementati su sistemi informatici e memorizzano i dati digitalmente.
- **Facilmente accessibile, gestita e aggiornata:** Un database è progettato per consentire agli utenti e alle applicazioni di interagire con i dati in modo controllato e sicuro. Questo include operazioni come l'inserimento, la lettura, la modifica e l'eliminazione dei dati.

Oltre alla semplice memorizzazione di file binari, un database include:

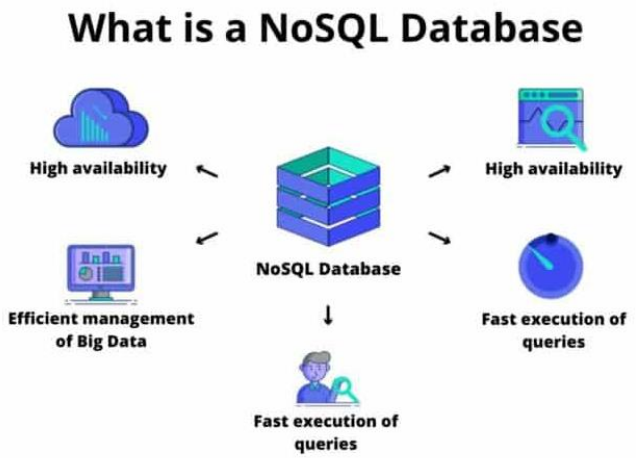
- **Schema del database:** La definizione della struttura del database, incluse le tabelle, le colonne, i tipi di dati, le relazioni tra le tabelle e i vincoli di integrità.
- **Sistema di gestione di database (DBMS):** Un software che permette di interagire con il database. Il DBMS fornisce funzionalità per definire, creare, interrogare, aggiornare e amministrare il database. Esempi di DBMS includono MySQL, PostgreSQL, Oracle, SQL Server e MongoDB.
- **Linguaggio di interrogazione:** Un linguaggio specifico (come SQL per i database relazionali) utilizzato per interagire con il database, recuperare informazioni e manipolare i dati.
- **Meccanismi di controllo e sicurezza:** Funzionalità per garantire l'integrità dei dati, la concorrenza degli accessi e la sicurezza contro accessi non autorizzati.
- **Indici e altre strutture di dati:** Ottimizzazioni per velocizzare le operazioni di ricerca e recupero dei dati.

In sintesi:
Un database è un sistema complesso che va ben oltre una semplice raccolta di file binari. È una struttura organizzata e gestita da un software specifico (DBMS) per archiviare, recuperare e manipolare dati in modo efficiente e affidabile. I file binari sono solo il modo in cui i dati vengono fisicamente memorizzati sul disco,

ma la vera essenza di un database risiede nella sua organizzazione logica, nella sua struttura e nelle funzionalità offerte dal DBMS.

Database NoSQL

I database NoSQL non hanno una data di nascita precisa come un singolo prodotto. Il termine "NoSQL" è emerso più chiaramente **alla fine degli anni 90 inizio 2000**, con la crescente popolarità di diverse tipologie di database che non aderivano al modello relazionale tradizionale e al linguaggio SQL.
Tuttavia, le idee e le tecnologie alla base dei database NoSQL hanno radici più antiche. Già negli anni '70 e '80 esistevano sistemi di gestione dati che non erano puramente relazionali, come i database gerarchici e di rete. Inoltre, alcuni sistemi di archiviazione chiave-valore possono essere considerati precursori dei moderni database NoSQL.

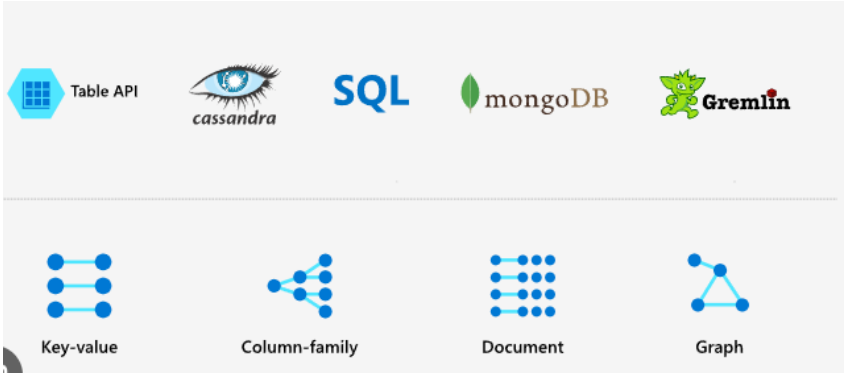


L'**esigenza principale** che ha portato alla ribalta i database NoSQL è stata la necessità di gestire le sfide poste dalle **applicazioni web moderne su larga scala** e dalla **crescente quantità e varietà di dati**. In particolare:

- **Scalabilità orizzontale:** I database relazionali tradizionali spesso faticavano a scalare orizzontalmente (aggiungendo più server) in modo efficiente ed economico per gestire l'enorme traffico e i volumi di dati generati dalle applicazioni web e dai social media. I database NoSQL sono stati progettati per essere distribuiti su più nodi, offrendo una migliore scalabilità orizzontale.
- **Flessibilità dello schema:** Le applicazioni moderne spesso lavorano con dati eterogenei non strutturati o semi-strutturati (come documenti JSON, XML, grafi), che non si adattano facilmente agli schemi rigidi dei database relazionali. I database NoSQL offrono schemi più flessibili e dinamici, consentendo di gestire diversi tipi di dati senza dover definire tabelle e colonne fisse in anticipo.
- **Alte prestazioni:** Per molte applicazioni web in tempo reale, è fondamentale avere tempi di risposta rapidi. Alcuni database NoSQL sono ottimizzati per operazioni di lettura e scrittura veloci, spesso sacrificando alcune delle proprietà ACID (Atomicità, Coerenza, Isolamento, Durabilità) tipiche dei database relazionali in favore di una maggiore disponibilità e tolleranza alle partizioni (secondo il teorema CAP).
- **Agilità nello sviluppo:** La flessibilità dello schema dei database NoSQL si adatta meglio ai cicli di sviluppo agili, in cui i requisiti dei dati possono cambiare rapidamente. Gli sviluppatori possono modificare la struttura dei dati senza dover eseguire migrazioni complesse dello schema.

In sintesi, i database NoSQL sono nati come risposta alle limitazioni dei database relazionali nel gestire le esigenze di scalabilità, flessibilità e prestazioni delle applicazioni web moderne e della gestione di grandi volumi di dati eterogenei.

Tipologie dei database NoSQL



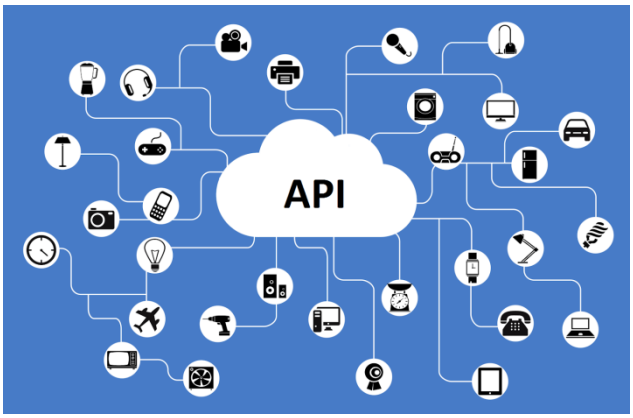
Esistono diverse tipologie di database NoSQL, ognuna ottimizzata per specifici casi d'uso e modelli di dati. Le principali categorie sono:

1. **Database Chiave-Valore (Key-Value Stores):**
    - Memorizzano i dati come coppie chiave-valore, dove ogni chiave è univoca e identifica un valore.
    - Sono altamente scalabili e offrono operazioni di lettura e scrittura molto veloci.
    - **Esempi:** Redis, Memcached, Amazon DynamoDB.
    - **Casi d'uso:** Caching, gestione delle sessioni utente, profili utente semplici.
2. **Database Orientati ai Documenti (Document Databases):**
    - Memorizzano i dati come "documenti", spesso in formato JSON o XML. Ogni documento è auto-descrittivo e può avere una struttura interna variabile.
    - Offrono flessibilità nello schema e sono adatti per dati semi-strutturati.
    - **Esempi:** MongoDB, Couchbase, Amazon DocumentDB.
    - **Casi d'uso:** Cataloghi di prodotti, gestione di contenuti, profili utente complessi.
3. **Database Orientati ai Grafi (Graph Databases):**
    - Memorizzano i dati come nodi (entità) e archi (relazioni) per rappresentare e interrogare dati altamente connessi.
    - Sono ideali per analizzare le relazioni tra i dati.
    - **Esempi:** Neo4j, Amazon Neptune.
    - **Casi d'uso:** Social network, sistemi di raccomandazione, rilevamento frodi, knowledge graph.
4. **Database a Colonne (Wide-Column Stores o Column-Family Databases):**
    - Memorizzano i dati in colonne anziché in righe. Le colonne correlate sono raggruppate in "famiglie di colonne".
    - Sono progettati per gestire enormi volumi di dati e offrono elevate prestazioni per query su specifiche colonne.
    - **Esempi:** Apache Cassandra, Apache HBase.
    - **Casi d'uso:** Serie temporali, big data analytics, archiviazione di dati di sensori.

È importante notare che alcuni database NoSQL sono **multimodello**, il che significa che supportano più di un tipo di modello di dati. Ad esempio, Azure Cosmos DB supporta documenti, chiavi-valore, grafi e colonne. La scelta del tipo di database NoSQL dipende dalle specifiche esigenze dell'applicazione, dal tipo di dati da gestire e dai requisiti di scalabilità e prestazioni.

Le applicazioni client- server si compongono di tre parti

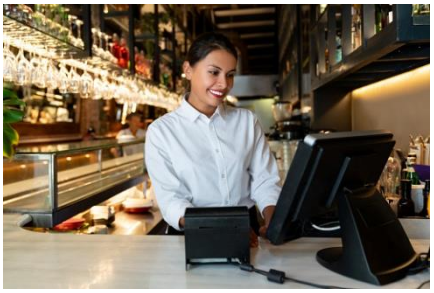
Il SERVER, il CLIENT e l' API

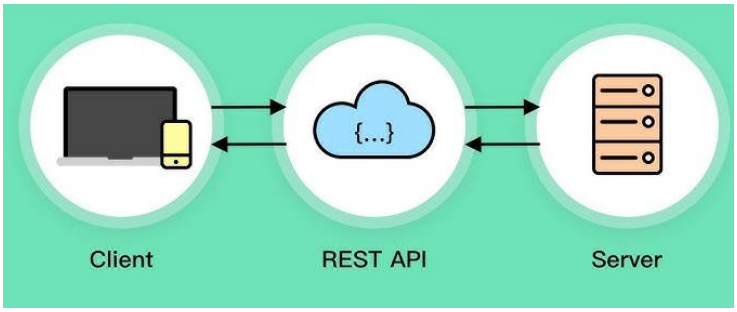


Viviamo in un mondo dove tutti sono interconnessi tra di loro, dove in pochi secondi riusciamo a mandare un messaggio ad una persona distante anche migliaia di kilometri e dove, comodamente dal letto con il nostro smartphone, potremmo anche programmare l'attivazione della macchina del caffè; ma **come avviene questa connessione**? Come fanno diversi dispositivi (tablet, pc, smartphone) e applicazioni a permetterlo?

L'eroe silenzioso e poco citato che permette tutto questo sono le **Application Programming Interface** o meglio conosciute come **API.**

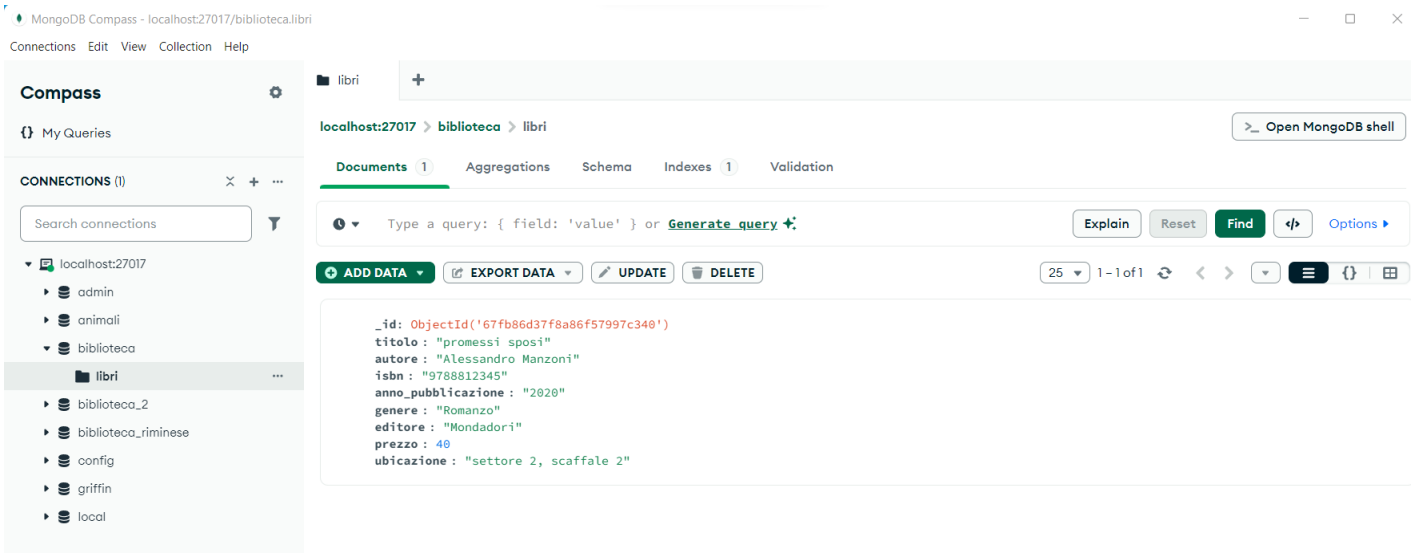
Immagina di andare al ristorante e di ordinare del cibo o chiedere informazioni in merito ad un piatto. Sicuramente, per fare una di queste azioni interagirai con un **cameriere,** rivolgendoti a lui anche per chiedere e pagare il conto oppure per altre informazioni sul menù.





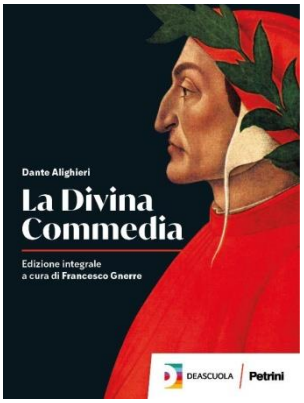
API

Il client MongoDB Compass



Il nostro programma client Database_Biblioteca realizzato in python con la libreria tkinter

I libri che inseriamo nel database




```
_id: ObjectId('67f77b397b9a283d5c1f83bb')
titolo: "promessi sposi"
autore: "alessandro manzoni"
anno_pubblicazione: 2025
genere: "romanzo"
isbn: 97888233799916
editore: "mondadori"
```

```
_id: ObjectId('67f77b397b9a283d5c1f83be')
titolo: "la divina commedia"
autore: "dante aligheri"
anno_pubblicazione: 2024
genere: "romanzo"
isbn: 97888233799917
editore: "deascuola-petrini"
```