Archivi, Basi di Dati e DBMS

Dall'approccio file-based a SQL

Prof. Francesco Gobbi

I.I.S.S. Galileo Galilei — Ostiglia (MN)

a.s. 2025/2026

Introduzione all'argomento

- Perché serve **memorizzare** i dati (persistenza)?
- Archivi a file: record, campi, organizzazione
- Limiti del file-based e perché nasce il **DBMS**
- Cenni su modelli (E/R, relazionale) e su **SQL**

Che cos'è un archivio

Un archivio non è semplicemente un insieme di dati, ma un insieme di **informazioni organizzate secondo un criterio logico** che permette di ritrovarle nel tempo.
Nasce per soddisfare un bisogno fondamentale: **ricordare in modo ordinato**.
Senza un archivio, i dati esistono solo mentre il programma è aperto.

La persistenza dei dati

Se chiudo un programma, i dati spariscono. Se spengo il computer, la RAM si svuota.
Gli archivi permettono ai dati di:

- rimanere nel tempo
- essere riletti e modificati
- essere consultati anche dopo giorni o anni

La persistenza dei dati: RAM vs Disco

RAM	HDD
RAM is a volatile memory that stores information or data temporarily for easy and instant access.	HDD is a non-volatile memory that is the main data storage hardware device in a computer.
The data is erased when the device is turned off.	The data remains even if the device loses power.
RAM is typically used to store working data and machine code.	HDD stores all your digital content and data on the computer.
RAM affects the speed of the computer.	HDD does not affect the speed of the computer.
RAM is smaller than HDDs.	HDDs are usually larger than RAM.
RAM has shorter read and write time.	HDDs have longer read and write time.

DB Difference Between.net

Record, campi e tuple: come sono organizzati i dati

Ogni archivio è composto da **record**. Ogni record descrive un elemento attraverso più **campi**.

Esempio concreto: una rubrica telefonica.

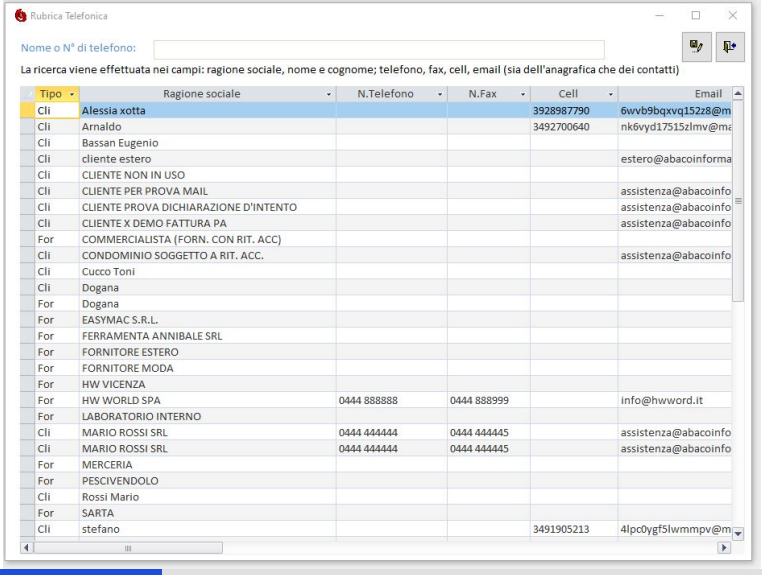
- Un **record** rappresenta una persona.
- I **campi** descrivono quella persona: nome, cognome, telefono.

Se immaginiamo questi dati in forma tabellare:

- i **campi** corrispondono alle **colonne** della tabella;
- ogni **record** corrisponde a una **riga** della tabella.

Nel modello relazionale, una riga della tabella prende il nome di **tupla**: una tupla è quindi l'insieme dei valori che descrivono una singola istanza (tali valori possono anche essere di tipologie diverse).

Record e campi: come sono organizzati i dati



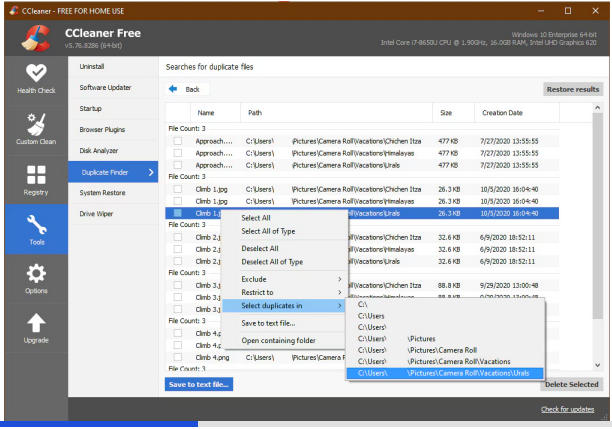
Tipo	Ragione sociale	N.Telefono	N.Fax	Cell	Email
Cli	Alessia xotta			3928987790	6wvb9bqxvq152z8@m
Cli	Arnaldo			3492700640	nk6vyd17515zlmv@ma
Cli	Bassan Eugenio				
Cli	cliente estero				estero@abacoinforma
Cli	CLIENTE NON IN USO				
Cli	CLIENTE PER PROVA MAIL				assistenza@abacoinfo
Cli	CLIENTE PROVA DICHIARAZIONE D'INTENTO				assistenza@abacoinfo
Cli	CLIENTE X DEMO FATTURA PA				assistenza@abacoinfo
For	COMMERCIALISTA (FORN. CON RIT. ACC)				
Cli	CONDOMINIO SOGGETTO A RIT. ACC.				assistenza@abacoinfo
Cli	Cucco Toni				
Cli	Dogana				
For	Dogana				
For	EASYMAC S.R.L.				
For	FERRAMENTA ANNIBALE SRL				
For	FORNITORE ESTERO				
For	FORNITORE MODA				
For	HW VICENZA				
For	HW WORLD SPA	0444 888888	0444 888999		info@hwword.it
For	LABORATORIO INTERNO				
Cli	MARIO ROSSI SRL	0444 444444	0444 444445		assistenza@abacoinfo
Cli	MARIO ROSSI SRL	0444 444444	0444 444445		assistenza@abacoinfo
For	MERCERIA				
For	PESCIVENDOLO				
Cli	Rossi Mario				
For	SARTA				
Cli	stefano			3491905213	4lpc0ygf5lwmmpv@m

Dal concetto di archivio al file

Nel computer, l'archivio viene memorizzato in un **file**.
Il file diventa il contenitore fisico dei record e permette di: inserire dati, modificarli, cercarli e cancellarli.
Questo approccio viene chiamato **file-based** ed il software che gestisce i file è il **file-system**, parte del Sistema Operativo.

Il problema dell'approccio file-based

Quando il sistema cresce, iniziano i problemi.
Gli stessi dati vengono salvati in più file. Diventa difficile mantenerli coerenti e aggiornati.
Questo prende il nome di **ridondanza**.



Perché nasce il database

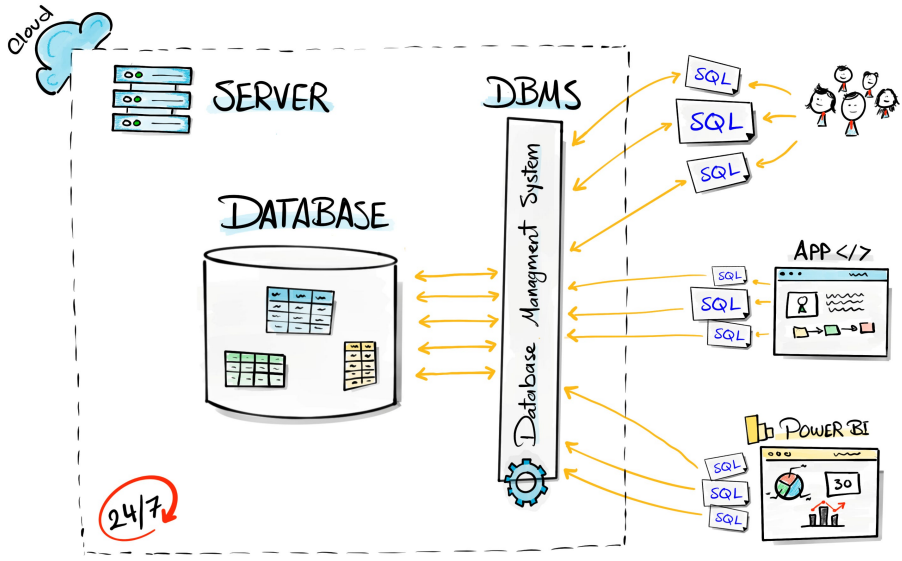
Quando i dati sono salvati in tanti file diversi, diventano difficili da gestire, aggiornare e mantenere coerenti.

Il **database** nasce per raccogliere tutte le informazioni in un unico sistema organizzato.

Questo permette di:

- **Evitare duplicazioni**: lo stesso dato viene salvato una sola volta.
- **Mantenere i dati coerenti**: ogni modifica è visibile ovunque serva.
- **Interrogare le informazioni in modo flessibile**: si possono fare ricerche non previste in anticipo.

Perché nasce il database



Il ruolo del DBMS

Il DBMS è il **software** che si occupa di gestire il database e di fare da intermediario tra chi usa i dati e dove i dati sono memorizzati.
Non si limita a salvare le informazioni, ma svolge funzioni fondamentali:

- **Integrità**: controlla che i dati inseriti rispettino regole e vincoli, evitando errori e incoerenze.
- **Sicurezza**: gestisce utenti e permessi, stabilendo chi può vedere o modificare determinate informazioni.
- **Gestione multiutente**: permette a più persone di lavorare sugli stessi dati contemporaneamente senza creare conflitti.
- **Protezione da errori e guasti**: attraverso backup, ripristino e gestione delle transazioni, evita la perdita di dati.

Dal problema reale al modello E/R o Modello Concettuale

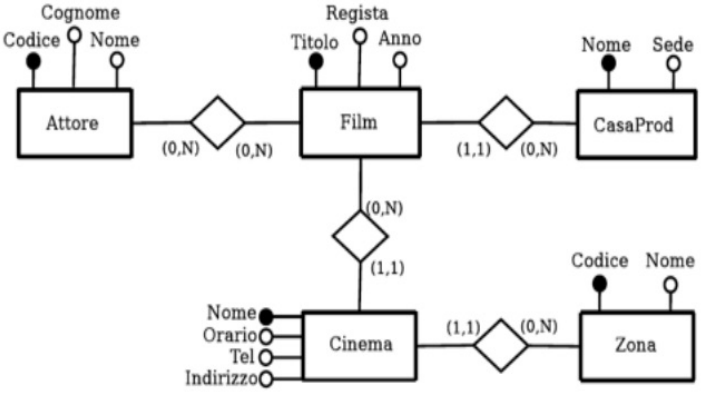
Nei database relazionali, quindi crati con relazioni, (come MySQL) i dati vengono salvati in **tabelle** (righe e colonne). La parte iniziale ed importante è cosa serve per creare un database, quindi capire cosa si vuole effettivamente salvare e rappresentare della realtà.

Il modello **E/R (Entity/Relationship)** serve proprio a questo: descrivere il dominio del problema in modo chiaro, senza pensare ancora ai dettagli tecnici.

- **Entità**: gli "oggetti" importanti del sistema informatico da realizzare (es. Studente, Libro, Prestito).
- **Attributi**: le proprietà delle entità (es. nome, data_nascita, titolo).
- **Relazioni**: definisce come le entità sono collegate tra loro (es. Studente *effettua* Prestito).

Riassunto: prima viene modellata la realtà con l'E/R, poi lo si trasforma in **tabelle** del database.

Esempio di Modello Concettuale



N.B. Questa notazione non verrà utilizzata per gli esercizi. In seguito vedremo la notazione per creare il modello E/R.
Solitamente questa notazione è molto utilizzata all'università.

Dal modello E/R al modello logico e poi fisico

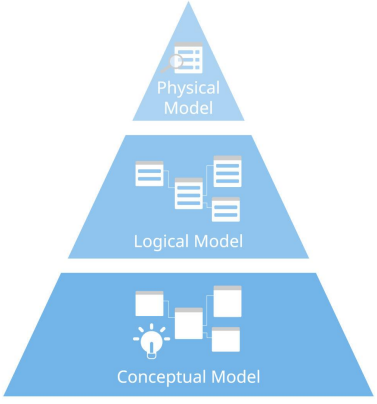
Il modello **E/R** è il **modello concettuale**: descrive la realtà del problema senza pensare ancora alle tabelle.

Quando si passa al **modello logico (chiamato anche modello logico-relazionale)**, si trasformano gli elementi del modello E/R in strutture tabellari.
In particolare:

- le **entità** diventano **tabelle**;
- gli **attributi chiave** dell'entità diventano la **chiave primaria** della tabella;
- le **relazioni** (in base alla loro cardinalità) determinano la presenza di **chiavi esterne** nelle tabelle.

Infine, il modello logico viene realizzato nel **modello fisico**, ad esempio creando concretamente le tabelle con il linguaggio **MySQL**.

Schema di creazione dei vari modelli



Le proprietà ACID e l'affidabilità del database

Quando più operazioni agiscono sui dati contemporaneamente, il rischio di errori diventa molto alto.
Per questo motivo, i database utilizzano il concetto di **transazione**: un insieme di operazioni che devono essere trattate come un unico blocco logico.

Affinché una transazione sia affidabile, deve rispettare quattro proprietà fondamentali, riassunte nell'acronimo **ACID**:

Atomicità **Consistenza** **Isolamento** **Durabilità**

Queste proprietà permettono al database di funzionare correttamente anche in presenza di errori, guasti o accessi simultanei.

Atomicità: tutto o niente

L'**atomicità** stabilisce che una transazione deve essere eseguita completamente oppure non essere eseguita affatto.

Non sono ammessi stati intermedi: se una sola operazione fallisce, tutte le operazioni precedenti vengono annullate automaticamente.

Esempio reale: durante un bonifico bancario, il denaro deve essere scalato dal conto del mittente e accreditato al destinatario. Se l'accredito fallisce, anche il prelievo viene annullato.

Questo evita che il database rimanga in uno stato incoerente.

Consistenza: i dati devono rimanere validi

La **consistenza** garantisce che il database, prima e dopo la transazione, rispetti sempre tutte le regole e i vincoli definiti.

I dati non possono mai trovarsi in una situazione non valida rispetto alla struttura del database.

Esempio reale: non è possibile registrare un prestito di un libro se lo studente non è presente nel database, oppure inserire un voto a uno studente inesistente.

Il DBMS controlla automaticamente che questi vincoli vengano rispettati.

Isolamento: lavorare insieme senza interferenze

L'**isolamento** permette a più utenti di operare contemporaneamente sugli stessi dati senza creare conflitti.

Ogni transazione viene eseguita come se fosse l'unica in corso, anche se in realtà il database sta gestendo molte operazioni insieme.

Esempio reale: due persone tentano di acquistare l'ultimo biglietto disponibile. Il sistema deve assegnarlo a uno solo, evitando che entrambi lo acquistino.

Questo è possibile grazie ai meccanismi di controllo della concorrenza.

Durabilità: ciò che è salvato resta salvato

La **durabilità** garantisce che, una volta confermata una transazione (operazione di *commit*), i dati restino memorizzati in modo permanente.

Anche in caso di guasti, spegnimenti improvvisi o errori hardware, le modifiche effettuate non vengono perse.

Esempio reale: dopo aver effettuato un acquisto online, l'ordine rimane registrato nel sistema anche se subito dopo salta la corrente.

Questo è garantito dai meccanismi di scrittura sicura su disco e dai log del DBMS.

Le proprietà ACID