Informatica: il trattamento automatico dell'informazione

Dopo l'invenzione della scrittura e la diffusione della stampa, il trattamento automatico delle informazioni appare oggi indiscutibilmente come la terza grande innovazione tecnologica destinata a introdurre trasformazioni radicali nel modo di produrre, manipolare e conservare le rappresentazioni simboliche del pensiero umano

http://www.treccani.it/enciclopedia/informatica_%28Enciclopedia_delle_scienze_sociali%29/

Informatica, computer, digitalizzazione

Informatica : da *Informa~~tion automa~~tique* (1962 Philippe Dreyfus)

Computer : calcolatore, elaboratore

Computer science : scienza del computer, Informatica

Digitale: da *Digit* (cifra, numero)

Dati e informazioni

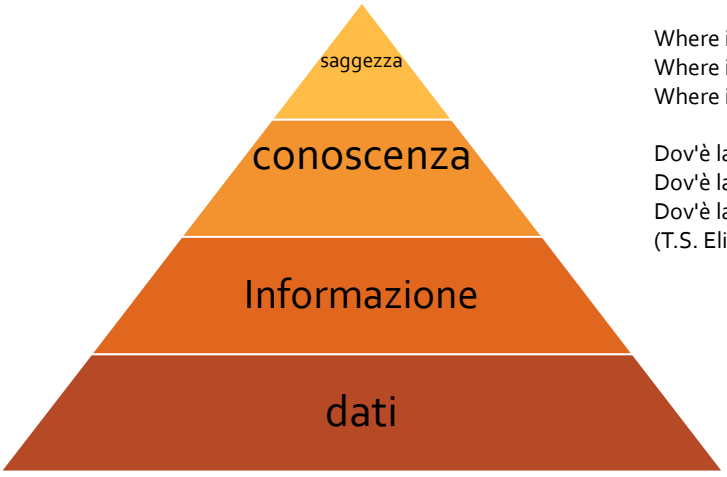
- Dato: un insieme di simboli che rappresente una proprietà di un oggetto nel mondo reale senza alcun riferimento alla proprietà stessa.

buono

- Informazione: un dato messo in relazione con la proprietà cui si riferisce

Stato di conservazione: buono

La piramide della conoscenza



Where is the Life we have lost in living?
Where is the wisdom we have lost in knowledge?
Where is the knowledge we have lost in information?

Dov'è la vita che abbiamo perso vivendo?
Dov'è la saggezza che abbiamo perso conoscendo?
Dov'è la conoscenza che abbiamo perso informandoci?
(T.S. Eliot, 1934 The Rock)

La codifica e la rappresentazione dell'informazione

un dato o un'informazione può essere codificata con simboli e modalità diverse

MCCCLVI = 1356

Nei calcolatori ogni informazione è rappresentata in forma numerica sulla base di una codifica binaria, e dunque attraverso bit (binary digit, cifra binaria)

Rappresentazione digitale dell'informazione

termine 'digitale' non si riferisce di norma solo al fatto che l'informazione è rappresentata in forma numerica, ma al fatto che è rappresentata in forma numerica sulla base di una *codifica binaria*, e dunque attraverso bit (il termine *bit* corrisponde alla contrazione dell'inglese ***bi****nary digi****t***, numero binario).

Bit e Byte

1 bit può assumere due configurazioni possibili

«0» o «1»

ma una sequenza di bit ....

- 2 bit: 4 ($=2^2$) sequenza possibili 00, 01, 10, 11
- 3 bit: 8 ($=2^3$) sequenze possibili 000, 001, 010, 011, 100, 110, 111
- .....
- 8 bit; 256 ($=2^8$) sequenze possibili 00000000, 00000001, 00000010, 00000011.....

Byte= sequenza di 8 bit

ogni numero decimale può essere trasformato in un numero binario (costruito usando solo lo '0' e l''1').

COME??

lo '0' e l''1' resteranno uguali, ma il 2 sarà rappresentato dalla combinazione '10', il 3 da '11', il 4 da '100', il 5 da '101', il 6 da '110', il 7 da '111', l'8 da '1000' e così via: la nostra rappresentazione dei numeri superiori a 2 sarà cioè ottenuta combinando fra loro (in maniera ordinata) un numero via via maggiore di '0' e di '1', proprio come nel nostro familiare sistema decimale i numeri superiori al 9 vengono costruiti combinando fra loro (in maniera ordinata) le dieci cifre che abbiamo a disposizione.

Numero decimale	Valore delle posizioni binarie				
	16	8	4	2	1
0	0	0	0	0	0
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	0	1	1
4	0	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	0	1	1	1
8	0	1	0	0	0
9	0	1	0	0	1
10	0	1	0	1	0
11	0	1	0	1	1
12	0	1	1	0	0
13	0	1	1	0	1
14	0	1	1	1	0
15	0	1	1	1	1
16	1	0	0	0	0

Dal testo ai numeri

testo è una *successione di caratteri*, i caratteri di base - quelli compresi nell'alfabeto della lingua usata - sono in un numero che varia col variare delle lingue, ma che è comunque - almeno per le lingue basate sull'alfabeto latino **- finito e piuttosto ristretto.**

tabella di corrispondenza fra caratteri da un lato e numeri binari dall'altro.

tabella di codifica dei caratteri.

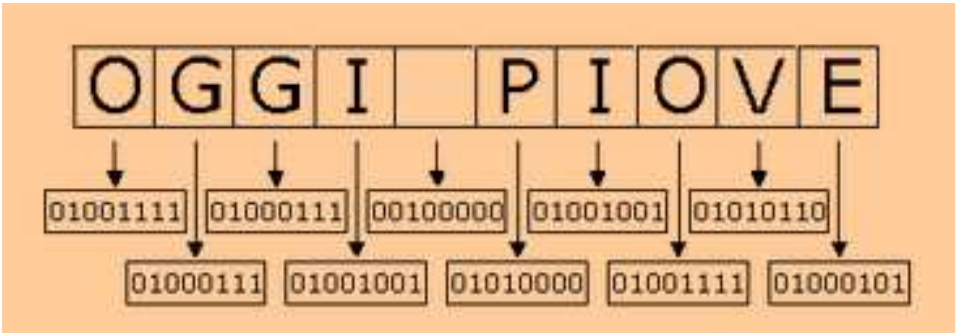
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	_
48	0	64	@	80	P	96	`	112	p		

Codifica ASCII e ISO Latin1

La codifica ASCII originaria (ASCII stretto) permetteva di distinguere 128 caratteri diversi

la tabella di caratteri ISO Latin 1, distingue 256 caratteri, i primi 128 dei quali sono 'ereditati' dall'ASCII stretto. L'indicazione ISO indica l'approvazione da parte dell'International Standardization Organization e 'Latin 1' indica che si tratta della tabella di riferimento per gli alfabeti di tipo latino. È questa la codifica di caratteri utilizzata dalla maggior parte dei sistemi operativi (Windows, Macintosh...). Come ogni tabella di codifica dei caratteri, anche la tabella ISO Latin 1 codifica i caratteri da essa

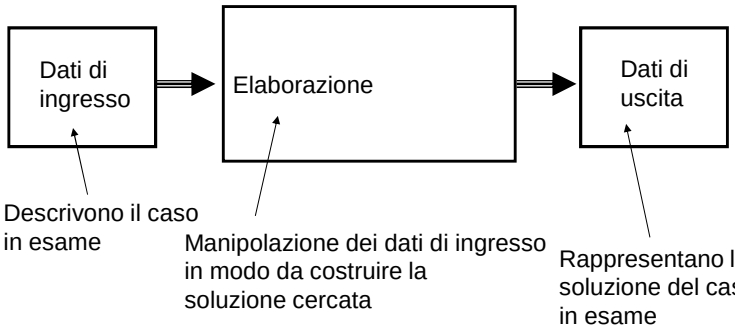
I numeri sono tutti espressi attraverso una notazione 'lunga' esattamente otto cifre binarie, ovvero 8 bit. In sostanza, possiamo 'scriverlo' utilizzando otto cellette affiancate, in ciascuna delle quali può esserci uno '0' o un '1' (nel caso considerato, le cellette conterranno tutte degli '1').


OGGI PIOVE
01001111
01000111
00100000
01001001
01010110
01000111
01001001
01010000
01001111
01000101

HD E SW

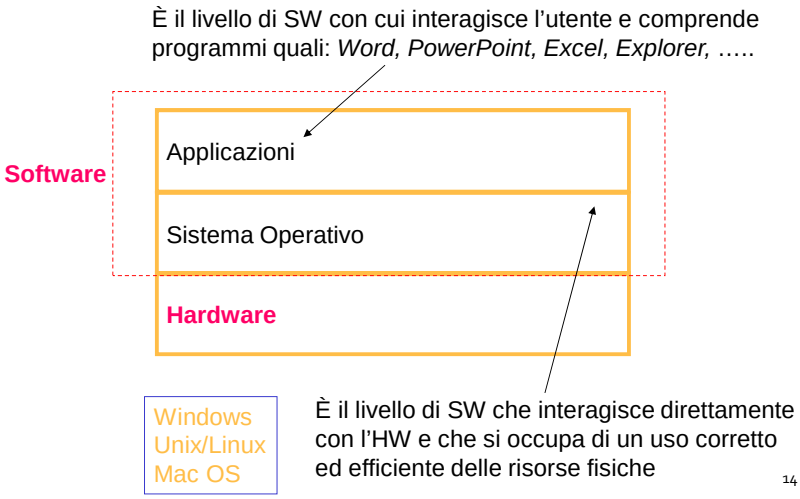
Il calcolatore come strumento per la manipolazione dell'informazione

- Come viene risolto un problema :



13

Componenti del computer



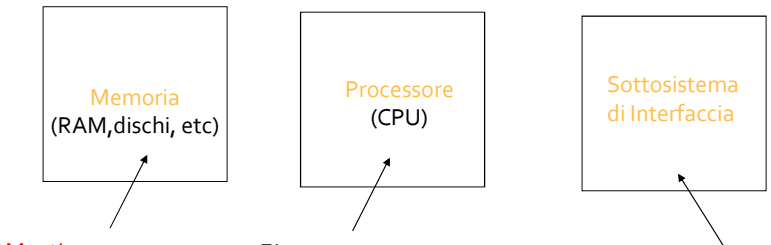
14

L'ARCHITETTURA DEL CALCOLATORE

15

Struttura di un calcolatore

- L'architettura di Von Neumann



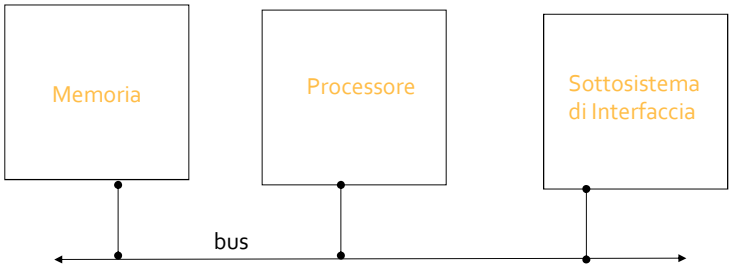
Mantiene
Dati e Programmi

E' un esecutore capace di interpretare i singoli passi richiesti dai programmi *(istruzioni elementari)*

Permette di comunicare dati e programmi alla macchina e di ottenere i risultati *(tastiera, micr., stampante, schermo,)*

16

Struttura di un calcolatore (2)

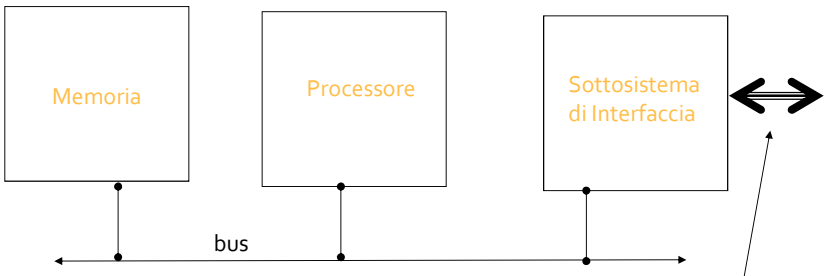


La comunicazione fra i tre sottosistemi viene effettuata attraverso un dispositivo fisico detto bus di interconnessione. Attraverso il bus la CPU

- legge\scrive dati e programmi in memoria
- trasferisce da\a la memoria al dispositivo di interfaccia
- recupera la prossima istruzione da eseguire

17

Struttura di un calcolatore (3)



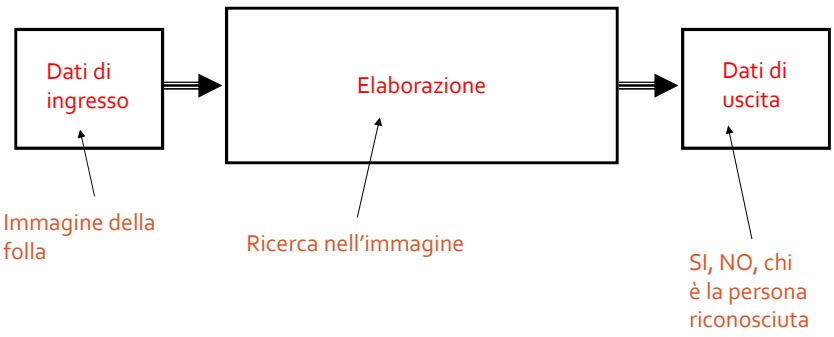
Il sottosistema di interfaccia cura anche la comunicazione fra due calcolatori diversi (es. via telefono\modem , via ethernet\ con opportune schede di connessione)

18

PROBLEMI, ALGORITMI E PROGRAMMI

Risolvere un problema

- es : riconoscere qualcuno fra la folla



Risolvere un problema

- es : torta di carote

21

Risolvere un problema

Essere capaci di risolvere un problema non significa essere capaci di spiegare esattamente come questo avviene

Come avviene la ricerca ??????

22

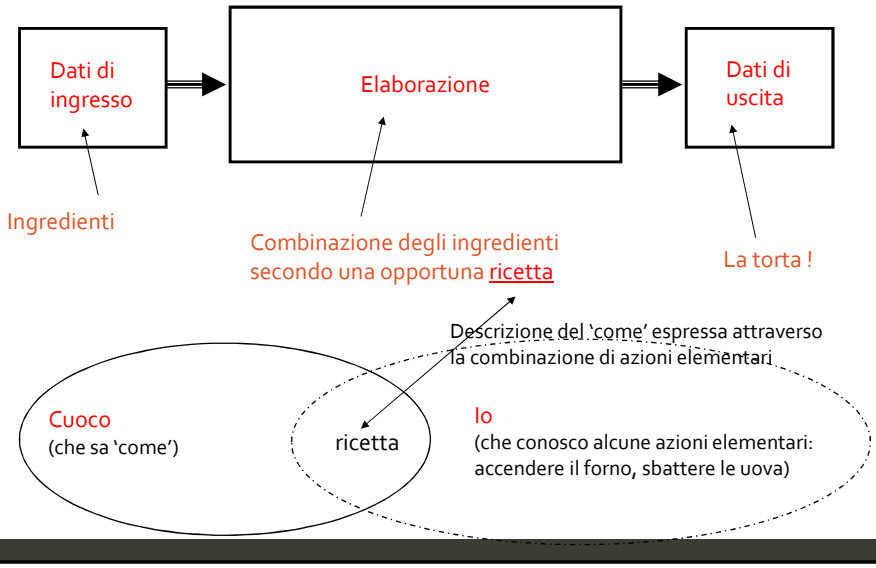
Essere capaci di risolvere un problema non significa essere capaci di spiegare esattamente come questo avviene

Se vogliamo essere capaci di specificare la strategia seguita dal passo di elaborazione in modo da farla eseguire 'automaticamente' dal computer quindi dobbiamo :

riuscire a descrivere accuratamente i vari passi della soluzione attraverso azioni che il calcolatore è in grado di effettuare e con un linguaggio che è in grado di comprendere

23

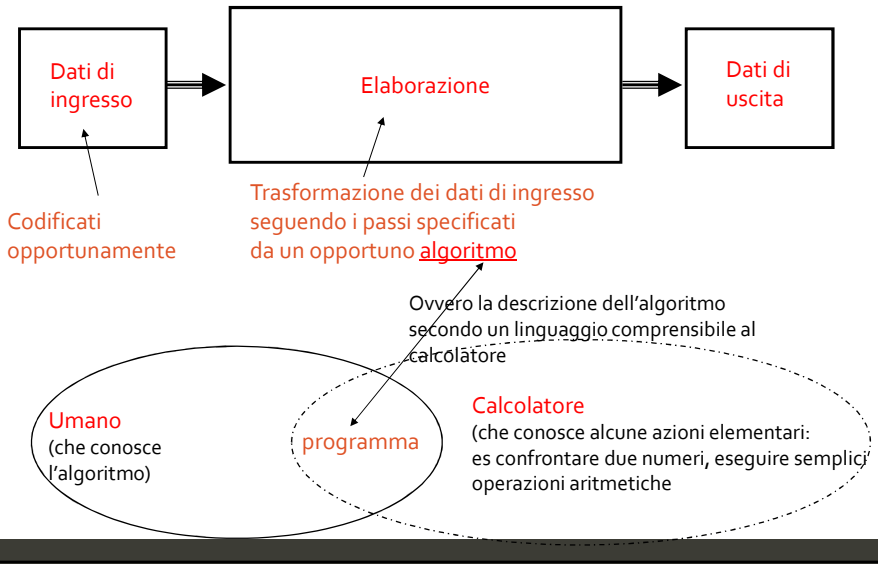
Risolvere un problema



24

Algoritmi e programmi

- La situazione con il calcolatore è



25

Algoritmi e programmi

- Algoritmo una sequenza di azioni *non ambigue* che trasformi i dati iniziali nel risultato finale utilizzando un insieme di azioni elementari che possono essere eseguite da un opportuno esecutore.
- Programma specifica di un algoritmo utilizzando un linguaggio non ambiguo e direttamente comprensibile dal computer

26

Algoritmi e programmi

- Due punti importanti:
  - fissare in modo meno vago le azioni elementari eseguibili da un computer
  - capire come passare dal problema all'algoritmo

27

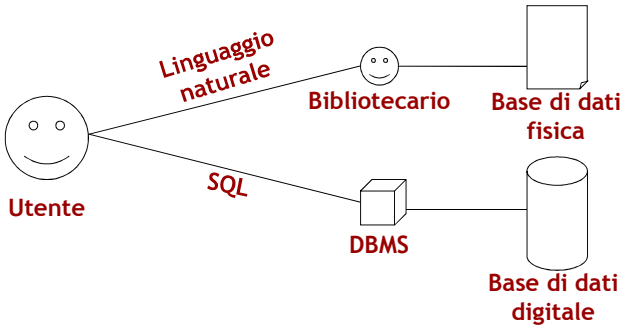
Algoritmi e programmi (4)

- Ma insomma, una ricetta è proprio un algoritmo?
  … quasi, ovvero è molto simile ad un algoritmo con due importanti differenze:
  - La sequenza di azioni contiene spesso degli elementi di *ambiguità* risolti da un esecutore intelligente
    - es: spesso non si specificano gli strumenti da utilizzare, confidando che l'esecutore umano *sbatta le uova* nel posto giusto
  - Non tutti i possibili casi vengono specificati
    - es: è chiaro che se c'e' puzza di bruciato conviene spegnere il forno, anche se la ricetta non lo specifica
    - anche qua si confida nelle capacità deduttive dell'esecutore

28

BASI DI DATI

Dall'utente alle informazioni



Vantaggi dei DB

- Risparmio di spazio: sono facilmente **trasferibili** e **duplicabili**
- Risparmio di tempo: si può accedere ai dati più rapidamente, per **consultarli** o **modificarli**
- Non si è vincolati ad un supporto **fisico**: questo permette di gestire diversi tipi di dati

Perché usare un DBMS?

•l'accesso ai dati è in**di**pendente dalla loro **rappresentazione** e **memorizzazione**;

•le tecniche di accesso ai dati sono ottimizzate, in maniera da migliorare la **performance** delle interrogazioni ai database;

•sono possibili controlli di **integrità** dei dati;

•sono possibili controlli di **accesso** ai dati;

•è possibile un accesso **multiutente**.

Modelli di dati

Una base di dati, per essere di qualche utilità, deve necessariamente avere una struttura, un modello di organizzazione.

Tra quelli esistenti, ricordiamo:

- il modello **gerarchico**;
- il modello a **oggetti** (object-oriented);
- il modello **relazionale**;
- il modello **relazionale a oggetti** (object-relational).

Cos'è una base di dati (database)?

E' un insieme di **archivi di dati**:

- ✓ organizzati in modo **integrato** attraverso tecniche di modellazione dei dati
- ✓ gestiti sulle memorie di massa dei computer attraverso **appositi software**

A cosa serve una base di dati ?

- Raggiungere una grande **efficienza** nel **trattamento** e nel **ritrovamento** dei dati,
- **Superare** i limiti presenti nelle organizzazioni tradizionali degli archivi quali:
  - **mancanza di integrazione** fra gli archivi;
  - **legame** fra l'**accesso** ai dati ed il tipo di **organizzazione** assegnata agli archivi.
  - **legame** fra gli **archivi ed i programmi,**

Quali problemi provocano gli archivi tradizionali?

Ridondanza

I dati sono **ridondanti** se compaiono in maniera duplicata (questo è un aspetto di inefficienza nella gestione).

Produce

Incongruenza

Un dato è **incongruente** se può essere aggiornato in un archivio e non in un altro, oppure possono essere presenti valori diversi per lo stesso dato.

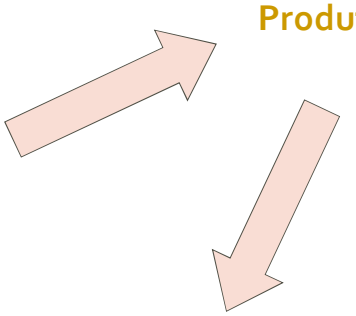
Produce

Inconsistenza

I dati sono **inconsistenti** se non sono più affidabili, perché non si sa in modo certo quale dei diversi valori sia quello corretto.

Cosa deve garantire un database?

Un database deve consentire di **ritrovare facilmente le informazioni desiderate** anche a fronte di una mole di dati rilevante.

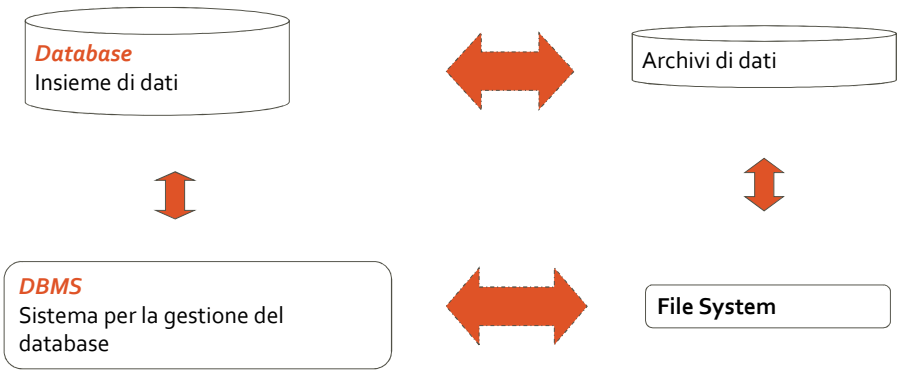


✓**velocità** nell'elaborazione,
✓**sicurezza** dei dati,
✓**integrità** delle registrazioni.

Database o DBMS ?

Nell'uso dei database ...

... come nell'uso degli archivi tradizionali



Il DBMS (Data Base Management System)

Accresce la differenza tra:

- **struttura concettuale:** modo attraverso il quale l'utente pensa all'organizzazione e al ritrovamento dei dati,
- **struttura fisica** dei dati: tecniche utilizzate dal Sistema Operativo per registrare e leggere negli archivi.

Come può il DBMS superare i limiti della gestione tradizionale?

Indipendenza dalla struttura fisica dei dati
I programmi applicativi sono indipendenti dai dati fisici.

Possibile modificare i supporti con cui i dati sono registrati e le modalità di accesso alle memoria di massa senza modifiche alle applicazioni.

Come può il DBMS superare i limiti della gestione tradizionale?

Indipendenza dalla struttura logica dei dati
I programmi applicativi sono indipendenti dalla struttura logica con cui i dati sono organizzati negli archivi

Possibile modificare la definizione delle strutture della base di dati senza modificare il software applicativo.

Come può il DBMS superare i limiti della gestione tradizionale?

Utilizzo da parte di più utenti

- I diversi utenti del database possono avere anche una **visione parziale** del database.
- Le **operazioni** svolte da utenti diversi in modo concorrente **non interferiscano una con l'altra.**

Come può il DBMS superare i limiti della gestione tradizionale?

Eliminazione di:

Ridondanza → Gli stessi dati non compaiono più volte in archivi diversi, cioè il database è costituito da **archivi integrati** di dati.

Inconsistenza → Il database non può presentare campi uguali con valori diversi in archivi diversi.

I modelli per il database

Nello sviluppo della teoria dei database, dal **1960** in poi, sono emersi tipi diversi di modelli per le basi di dati:

- **gerarchico,**
- **reticolare,**
- **relazionale,**
- **database orientati agli oggetti.**

Il modello relazionale

- Rappresenta il **database** come un **insieme di tabelle**.
- Viene considerato attualmente il **modello più semplice ed efficace**, perché è più vicino al modo consueto di pensare i dati, e si adatta in modo naturale alla classificazione e alla strutturazione dei dati.

Modello relazionale: concetti di base

Il modello relazionale si basa sul concetto matematico di **relazione** tra insiemi di oggetti.

Cos'è una relazione?

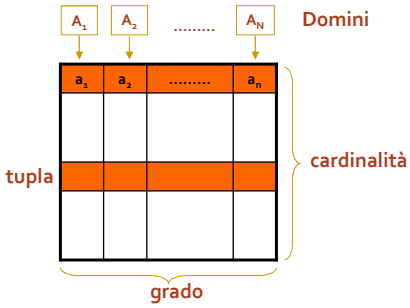
Definizione matematica

Dati *n* insiemi A_1, A_2,..., A_n, si dice **relazione** un sottoinsieme dell'insieme di tutte le n-uple a_1, a_2, ..., a_n che si possono costruire prendendo nell'ordine un elemento a_1 dal primo insieme A_1, a_2 dal secondo insieme A_2, e così via.

n = **grado** della relazione
A_i = **dominio** i-esimo della relazione
a_1, ... a_n = **tupla**
L'insieme delle tuple si chiama **cardinalità** della relazione

Modello relazionale: concetti di base

La relazione è rappresentata con una tabella, avente tante colonne quanti sono i domini (**grado**) e tante righe quante sono le n-uple (**cardinalità**).



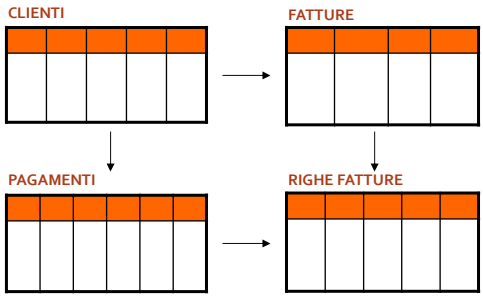
I nomi dei domini sono i nomi delle colonne, i valori che compaiono in una colonna sono **omogenei** tra loro (appartengono allo stesso dominio).

La relazione (*collezione di tuple*) è un'**entità**, ogni tupla è un'**istanza** dell'entità, le colonne sono gli **attributi** dell'entità, il dominio è l'insieme dei **possibili valori** di un attributo.

II/III

Modello relazionale: concetti di base

La **chiave** della relazione è un attributo o una combinazione di attributi che identificano univocamente le tuple: ogni riga della tabella possiede valori diversi per l'attributo (o gli attributi) chiave.



Il modello relazionale di un database è un **insieme di tabelle**, sulle quali si possono effettuare operazioni e tra le quali possono essere stabilite delle associazioni.

III/III

Modello relazionale: requisiti

a. tutte le righe della tabella contengono lo **stesso numero di colonne**;

b. gli attributi rappresentano **informazioni elementari** (*atomiche*), ovvero non scomponibili ulteriormente;

c. i valori assunti da un campo appartengono al dominio dei valori possibili per quel campo, ovvero sono **omogenei** tra loro;

d. in una relazione, ogni riga è diversa da tutte le altre, ovvero esiste un attributo o una combinazione di attributi che **identificano univocamente** la n-upla;

e. le n-uple compaiono nella tabella secondo un **ordine non prefissato**, ovvero non è rilevante il criterio con il quale le righe sono sistemate nella tabella.

Integrità sull'entità

Ogni dato elementare contenuto nel modello relazionale deve essere accessibile attraverso la combinazione di:
- nome della tabella,
- nome e valore della chiave,
- nome della colonna contenente il dato.

Nessuna componente della chiave primaria di una tabella può avere valore nullo.

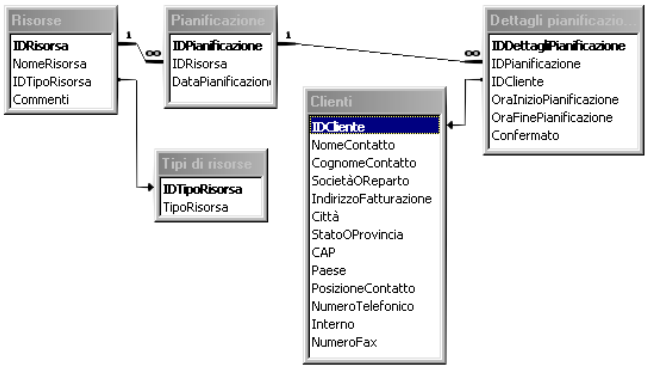
Le regole di derivazione del modello logico

Le tabelle vengono ricavate dal modello E/R applicando le **regole di derivazione**:

1. ogni entità diventa una relazione;
2. ogni attributo di un'entità diventa un attributo della relazione;
3. ogni attributo della relazione eredita le caratteristiche dell'attributo dell'entità da cui deriva;
4. l'identificatore univoco di un'entità diventa la ***chiave primaria*** della relazione derivata;
5. l'associazione ***uno a uno*** diventa un'unica relazione, che contiene gli attributi della prima e della seconda entità;
6. nell'associazione ***uno a molti***, l'identificatore univoco dell'entità di partenza diventa chiave esterna (***foreign key***) dell'entità di arrivo associata;
7. l'associazione con grado ***molti a molti*** diventa una nuova relazione, composta dagli identificatori univoci delle due entità e dagli eventuali attributi dell'associazione.

Relazioni e basi di dati relazionali

Una base di dati può essere formata da più tabelle non indipendenti. Tra di esse sono stabilite le relazioni tra tabelle.



Relazioni e basi di dati relazionali

Un concetto fondamentale nelle relazioni è il concetto di chiave.

Un insieme di attributi è detto *chiave* di una relazione se soddisfa alle seguenti condizioni:

- i valori non sono mai nulli
- identifica univocamente una ennupla
- non è ridondante, cioè ciascuno dei suoi valori è unico

La chiave può essere costituita da un solo attributo o può essere una *chiave multipla*.
Una relazione può avere più chiavi distinte.

Relazioni e basi di dati relazionali

Ogni tabella è caratterizzata da una e da una sola *chiave primaria*, cioè una chiave che caratterizza la tabella ed è opportunamente scelta per metterla in relazione con le altre tabelle.

Un attributo (o insieme di attributi) di una tabella T' è *chiave esterna* (con riferimento ad un'altra tabella T) se è *chiave primaria* di T

Una relazione fra due tabelle T e T' si stabilisce tra due attributi *a* (chiave primaria) di T e *b* di T' (chiave esterna) in modo che si corrispondano.
Gli attributi *a* e *b* devono essere omogenei e contenere coppie di valori uguali.

Relazioni tra tabelle

Fra due tabelle possono esiste i seguenti tipi di relazioni:

- *Relazione uno a uno*
- *Relazione uno a molti*
- *Relazione molti a molti*

Progettazione di una base di dati

La progettazione si occupa della costruzione dello schema della base di dati, cioè dei metadati.

La creazione di un database può essere suddivisa in tre fasi:

1. *ANALISI DEI REQUISITI: si stabilisce cosa si vuole rappresentare esattamente;*
2. *PROGETTO DEL SISTEMA: si progetta un sistema basandosi sulle informazioni ricavate dall'analisi dei requisiti;*
3. *REALIZZAZIONE DEL SISTEMA: si utilizza un DBMS (DataBase Management System).*

Gestione della base di dati

Dopo la fase di progettazione e realizzazione la base di dati può essere utilizzata dall'utente finale.
Le principali attività che si possono effettuare con un database sono:

- Caricamento dei dati
- Elaborazione del DB
- Transazioni applicative
- Interrogazioni
- Elaborazioni occasionali
- Riorganizzazione
- Salvataggio periodico
- Protezione del DB

Metadati

I METADATI sono lo schema della base di dati:

- definizioni che descrivono la struttura dei dati ;
- restrizioni sui valori ammissibili dei dati (vincoli di integrità) ;
- relazioni esistenti fra gli insiemi di dati (tabelle);
- operazioni eseguibili sui dati.

Dati

I DATI sono le informazioni memorizzate nell'archivio.

Principali caratteristiche:

- sono organizzati in insiemi omogenei (tabelle), fra i quali sono definite delle relazioni;
- sono in genere molti e non possono essere gestiti tutti contemporaneamente nella memoria temporanea (RAM);
- sono permanenti: una volta creati continuano ad esistere finché non vengono esplicitamente rimossi;
- sono condivisibili: più utenti possono utilizzarli anche nello stesso momento.

Relazione uno a uno

Fra due tabelle T e T' esiste una relazione uno a uno se per ciascun valore della chiave primaria *a* di T esiste al più un valore uguale della chiave esterna *b* di T'.

La relazione uno a uno è una relazione banale : indica che due tabelle potrebbero essere una tabella unica

Relazione uno a molti

Fra due tabelle T e T' esiste una relazione uno (T) a molti (T') se per ciascun valore della chiave primaria *a* di T possono esistere più valori uguali della chiave esterna *b* di T'.

La relazione uno a molti è la relazione più comune.
In una relazione uno-a-molti un record della tabella T può avere molti record corrispondenti nella tabella T', ma un record della tabella T' non ha più di un record corrispondente nella tabella T.

Relazione molti a molti

Fra due tabelle T e T' esiste una relazione molti a molti se per ciascun valore della chiave primaria *a* di T esistono più valori uguali della chiave esterna *b* di T' e viceversa.

In una relazione molti-a-molti un record della tabella T può avere molti record corrispondenti nella tabella T' e viceversa. Questo tipo di relazione è possibile solo definendo una terza tabella, chiamata tabella di congiunzione, la cui chiave primaria consiste almeno di due campi, vale a dire le chiavi esterne di entrambe le tabelle T e T'.

Chiave primaria

Ogni tabella deve avere una chiave primaria. Questa è un campo che non contiene ripetizioni e che rappresenta ogni record nelle altre tabelle.
Se non esiste un campo con tali caratteristiche, se ne deve aggiungere un nuovo campo che svolga la funzione di rappresentare univocamente ogni record. Solitamente un campo di tipo Contatore assolve a questa funzione.

Data	Quantità	Prezzo	IdCliente	Pagamento
21/12/02	100	L. 50.000	1	contanti
21/12/02	150	L. 35.000	2	contanti
21/12/02	100	L. 35.000	5	rateale
21/12/02	150	L. 50.000	25	bonifico
30/12/02	100	L. 50.000	1	contanti

Chiave primaria

IdOrdine	Data	Quantità	Prezzo	IdCliente	Pagamento
1	21/12/02	100	L. 50.000	1	contanti
2	21/12/02	150	L. 35.000	2	contanti
3	21/12/02	100	L. 35.000	5	rateale
4	21/12/02	150	L. 50.000	25	bonifico
5	30/12/02	100	L. 50.000	1	contanti

Campo con un solo valore

PRIMA FORMA NORMALE O FORMA ATOMICA:
ogni campo deve contenere un solo valore.
I principali motivi per cui non si devono avere campi che contengono più di un valore sono:

- i database relazionali non sono in grado di fare ricerche efficienti se il campo considerato contiene più di un valore
- le relazioni non possono esistere su campi contenenti più valori

IdPersona	Cognome	Nome	Indirizzo	Città
1	Rossi	Mario	via Roma, 4	PADOVA
2	Bianchi	Marco	via Trieste, 12	TREVISO
3	Verdi	Mara	via Aurora, 32	VENEZIA

IdPersona	Cognome	Nome	Indirizzo	Nome	Numero	Città
1	Rossi	Mario	via	Roma,	4	PADOVA
2	Bianchi	Marco	via	Trieste	12	TREVISO
3	Verdi	Mara	piazza	Aurora	32	VENEZIA

Indipendenza dei campi

SECONDA FORMA NORMALE:

tutti i campi, diversi dalla chiave primaria, devono dipendere solo dal valore della chiave primaria.

I campi delle tabelle non devono dipendere da altri campi, esclusa la chiave primaria. Questo garantisce che la chiave primaria rappresenti in modo univoco ogni record della tabella.

Le tabelle non possono contenere valori calcolati.
Per avere valori calcolati si devono utilizzare le query.

Ripetizione dei dati

Quando in una tabella i valori di alcuni campi sono ripetuti, è necessario creare due nuove tabelle che contengono i campi della prima. Le nuove tabelle devono poi essere messe in relazione tra di loro.

Cliente	Indirizzo	Citta	DataOrdin	Importo
Rossi	Via Roma, 4	Lecce	12-lug	€ 7.500,00
Rossi	Via Roma, 4	Lecce	15-ago	€ 5.478,00
Rossi	Via Roma, 4	Lecce	17-ago	€ 4.700,00
Rossi	Via Roma, 4	Lecce	12-set	€ 7.500,00
Rossi	Via Roma, 4	Lecce	15-ott	€ 5.478,00
Rossi	Via Roma, 4	Lecce	17-ott	€ 4.700,00
Rossi	Via Roma, 4	Lecce	12-nov	€ 7.500,00
Rossi	Via Roma, 4	Lecce	15-nov	€ 5.478,00
Rossi	Via Roma, 4	Lecce	17-dic	€ 4.700,00

Clienti

Ditta	Indirizzo	Città
Rossi	via Roma, 4	Lecce

Ordini

DataOrdine	Ordine
12-lug	7.500.000
15-ago	5.478.000
17-ago	4.700.000
12-set	7.500.000
15-ott	5.478.000

Analisi dei requisiti

L'analisi dei requisiti è lo studio preliminare che si deve fare prima di creare la base di dati, per stabilire:

- gli obiettivi dell'archivio
- i "campi di interesse"
- gli oggetti da rappresentare
- le azioni che si vorranno svolgere sull'archivio
- la durata che avrà l'archivio
- l'ambito in cui dovrà lavorare la base di dati
- tutto ciò che riguarda l'archivio stesso.

Progetto del sistema

A partire dal risultato dell'analisi dei requisiti, si struttura l'archivio, definendo:

- gli insiemi omogenei di dati (tabelle),
- i "campi di interesse" di ogni insieme (colonne delle tabelle),
- le caratteristiche di ogni campo (si tratta di un testo, di una data, di un numero, ecc.)
- le relazioni tra gli insiemi (i collegamenti tra le tabelle).

Il risultato di questa fase è lo schema della base di dati, "rappresentato" con carta e penna.
Questo schema è indipendente dal DBMS che poi sarà utilizzato per realizzare il database.

Realizzazione del sistema

La realizzazione del sistema progettato è effettuata con un *DataBase Management System* (ad esempio Access). La scelta del DBMS è determinata dalle esigenze degli utenti e dalla struttura che si vuole dare ai dati.

La realizzazione consiste nel partire dallo schema della base di dati realizzato nella fase precedente e crearlo, inserirlo, nel computer (si creano i metadati). La successiva fase sarà quella dell'inserimento dei dati.

Caricamento iniziale dei dati

E' la fase preliminare di registrazione di dati. Essa consiste nell'immissione delle ennuple definite dalla relazione che si accodano a quelle già esistenti.

Durante l'immissione vengono effettuati controlli di coerenza e compatibilità sui dati detti "*verifiche dei vincoli di integrità del database*".

E' possibile automatizzare l'immissione iniziale con tecniche basate u lettori ottici.

Elaborazione ordinaria del database

La fase di gestione ordinaria di un database consta di due classi fondamentali di operazioni:

- ✓ interrogazione, che non altera lo stato della base di dati (query);
- ✓ aggiunta, modifica, aggiornamento e cancellazione dei dati.

Per queste ultime, analogamente al caricamento iniziale, sono effettuate verifiche dei vincoli di integrità.

Transazioni applicative

Nella gestione di un database vengono definite alcune *transazioni*, cioè operazioni che alterano lo stato del database, per mezzo di procedure (programmi) che agiscono su di esso in modalità *batch* (automatica) o *interattiva*.

Questi programmi possono essere generati da appositi software associati al DBMS, o scritti in linguaggi di programmazione general purpose (*cobol, C++, ecc.*) nei quali è incorporato SQL (*Structured Query Language*).

Interrogazione della Base di Dati

Oltre alle interrogazioni programmate, se ne possono effettuare di estemporanee. In questi casi l'interrogazione del DB avviene attraverso linguaggi visuali di facile impiego direttamente dall'utente autorizzato alla selezione di informazioni che lo interessano.
In Access ci si riferisce alle *funzioni di trova* e ai *filtri*.

Elaborazioni occasionali

Esistono alcune operazioni occasionali o periodiche, quali la visita completa di una tabella con relativa elaborazione, la *stampa di tabulati* per controlli o verifiche, la *produzione di elenchi* completi o parziali (ad esempio la verifica di un anagrafico oppure quella delle giacenze di magazzino).
Sono queste tipiche operazioni eseguite dal DBA (*Data Base Administrator*).

Tipo di dati testo

Accetta dati lunghi fino a 255 caratteri, incluse combinazioni di testo e numeri, come nomi e indirizzi. Si usa questo tipo di dati per rappresentare numeri che non richiedono calcoli, come i numeri telefonici o i codici di avviamento postale.

Tipo di dati memo

Accetta dati lunghi fino a 65535 caratteri, incluse combinazioni di testo e numeri, commenti e lunghe descrizioni.

Tipo di dati numerico

Accetta valori numerici da usare nei calcoli.

Dimensione campo	Intervallo	Posizioni decimali
Byte	da 0 a 255	nessuna
Intero	da -32.768 a 32.767	nessuna
Intero lungo	da $-2{,}1\times10^{9}$ a $2{,}1\times10^{9}$	nessuna
Precisione singola	da $-3{,}4\times10^{38}$ a $3{,}4\times10^{38}$	7
Precisione doppia	da $-1{,}8\times10^{308}$ a $1{,}8\times10^{308}$	15
Decimale	da -10^{28} a 10^{28}	28

Tipo di dati data/ora

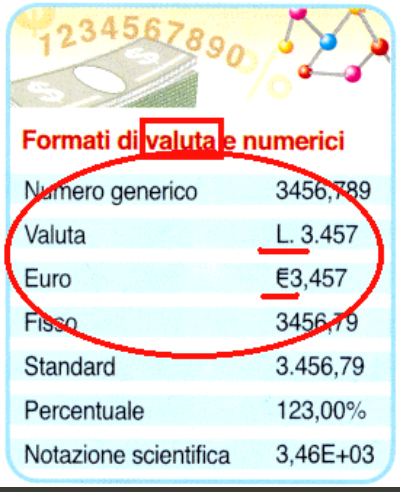
Accetta solo valori di data e ora.

Formati di data e ora

Data generica	19/06/99 17.34.23
Data estesa	sabato 19 giugno 1994
Data breve	19-giu-99
Data in cifre	19/06/94
Ora estesa	17.34.23
Ora breve 12h	5.34 PM
Ora breve 24h	17.34

Tipo di dati valuta

Accetta solo dati per valuta.



Tipo di dati Contatore

Numera automaticamente ogni record inserito.

Viene utilizzato per il campo chiave quando nessun altro campo soddisfa alle condizione di essere non vuoto e individuare univocamente i record.

Tipo di dati Si/No

Accetta solo uno dei due valori nei formati:

Si/No

Vero/Falso

On/Off

Tipo di dati OLE

Accetta solo oggetti OLE. Un oggetto OLE è creato con un'altra applicazione: può essere un documento, in foglio elettronico, un'immagine, ecc.

Tipo di dati
collegamento ipertestuale

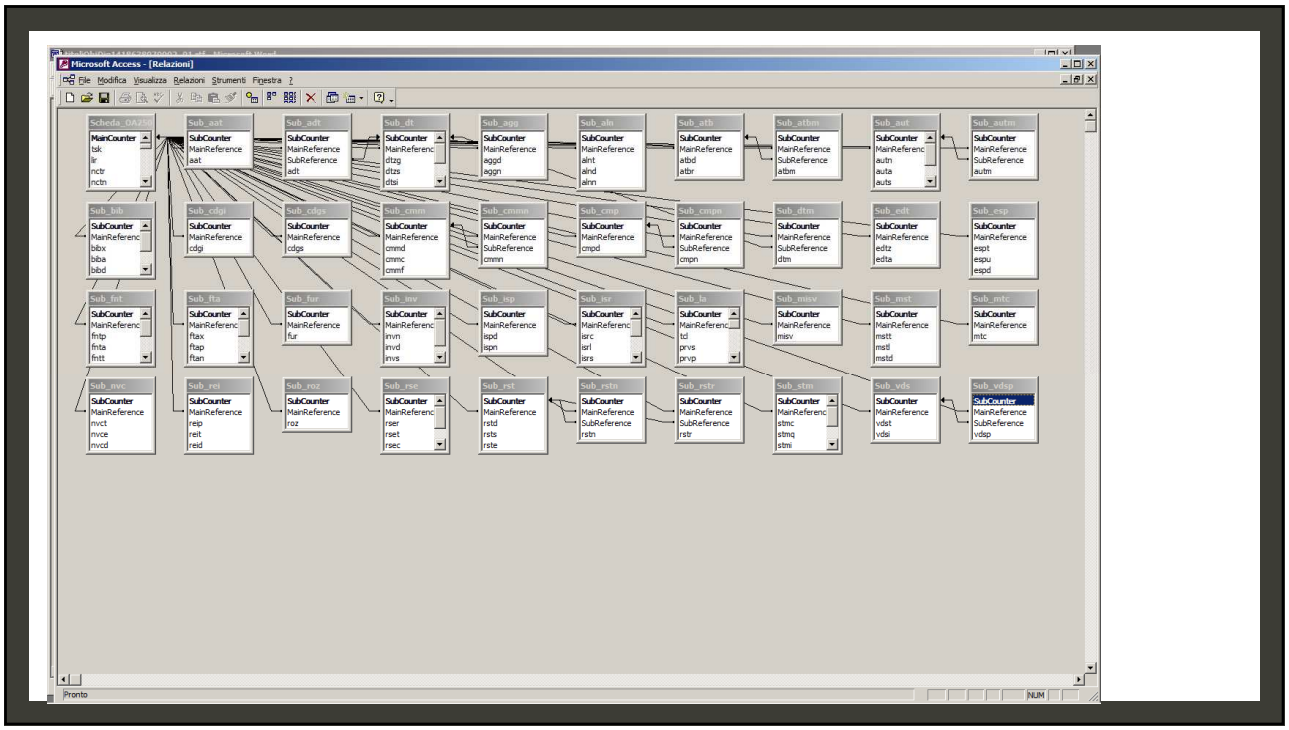
Accetta collegamenti ipertestuali da selezionare per accedere ad un documento o a una pagina Web..

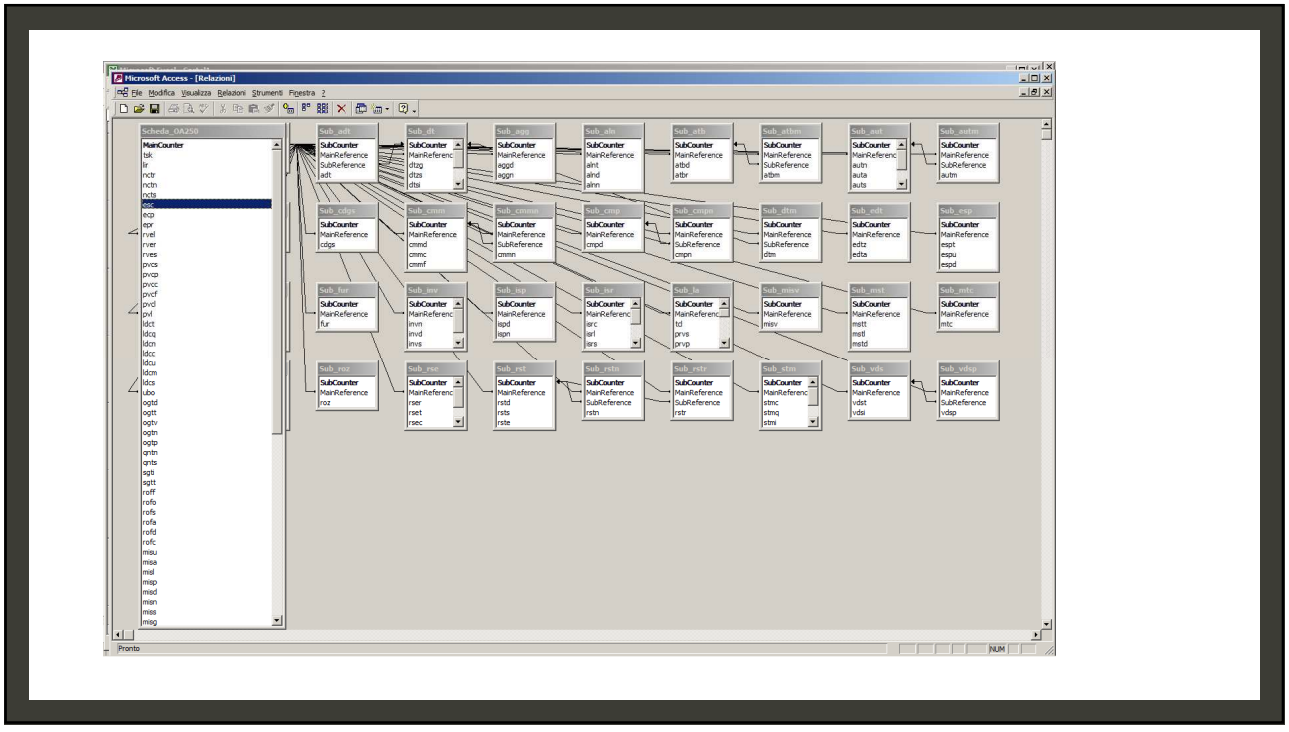
DBMS relazionali

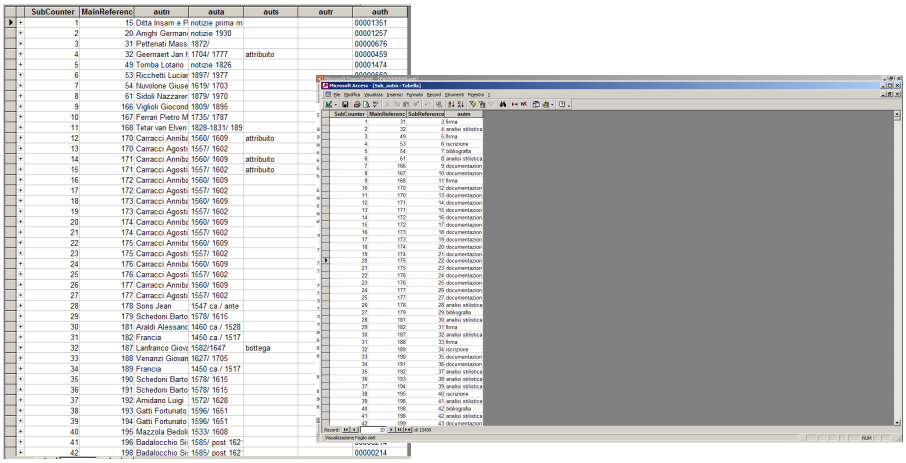
Nel modello relazionale, un database è un insieme di **relazioni**. Ciascuna di esse consiste di una **tabella** (righe e colonne).

Le colonne di una tabella sono anche dette **campi**, o **attributi** della tabella.

Ogni riga costituisce invece un **record**.


Microsoft Access - [Relazioni]
File Modifica Visualizza Relazioni Strumenti Finestra ?
Pronto


Microsoft Access - [Relazioni]
File Modifica Visualizza Relazioni Strumenti Finestra ?
Pronto



SubCounter	MainReferenc	autn	auta	auts	autr	auth
1	15	Ditta Insam e P	notizie prima m			00001351
2	20	Amighi Germani	notizie 1930			00001257
3	31	Pettenati Mass	1872/			00000676
4	32	Geemaert Jan I	1704/ 1777	attribuito		00000459
5	49	Tomba Lotario	notizie 1826			00001474
6	53	Ricchetti Luciar	1897/ 1977			00000560
7	54	Nuvolone Giuse	1619/ 1703			
8	61	Sidoli Nazzarer	1879/ 1970			
9	166	Viglioli Giocond	1809/ 1895			
10	167	Ferrari Pietro M	1735/ 1787			
11	168	Tetar van Elven	1828-1831/ 189			
12	170	Carracci Annibz	1560/ 1609	attribuito		
13	170	Carracci Agosti	1557/ 1602			
14	171	Carracci Annibz	1560/ 1609	attribuito		
15	171	Carracci Agosti	1557/ 1602	attribuito		
16	172	Carracci Annibz	1560/ 1609			
17	172	Carracci Agosti	1557/ 1602			
18	173	Carracci Annibz	1560/ 1609			
19	173	Carracci Agosti	1557/ 1602			
20	174	Carracci Annibz	1560/ 1609			
21	174	Carracci Agosti	1557/ 1602			
22	175	Carracci Annibz	1560/ 1609			
23	175	Carracci Agosti	1557/ 1602			
24	176	Carracci Annibz	1560/ 1609			
25	176	Carracci Agosti	1557/ 1602			
26	177	Carracci Annibz	1560/ 1609			
27	177	Carracci Agosti	1557/ 1602			
28	178	Sons Jean	1547 ca / ante			
29	179	Schedoni Barto	1578/ 1615			
30	181	Araldi Alessanc	1460 ca / 1528			
31	182	Francia	1450 ca / 1517			
32	187	Lanfranco Giov	1582/1647	bottega		
33	188	Venanzi Giovan	1627/ 1705			
34	189	Francia	1450 ca / 1517			
35	190	Schedoni Barto	1578/ 1615			
36	191	Schedoni Barto	1578/ 1615			
37	192	Amidano Luigi	1572/ 1628			
38	193	Gatti Fortunato	1596/ 1651			
39	194	Gatti Fortunato	1596/ 1651			
40	195	Mazzola Bedoli	1533/ 1608			
41	196	Badalocchio Si	1585/ post 162			
42	198	Badalocchio Si	1585/ post 162			00000214

SQL

SQL (**Structured Query Language**) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS) progettato per:

- creare e modificare schemi di database (DDL - *Data Definition Language*);
- inserire, modificare e gestire dati memorizzati (DML - *Data Manipulation Language*);
- interrogare i dati memorizzati (DQL - *Data Query Language*);
- creare e gestire strumenti di controllo ed accesso ai dati (DCL - *Data Control Language*)

non si tratta di un semplice linguaggio di interrogazione, ma alcuni suoi sottoinsiemi si occupano della creazione, della gestione e dell'amministrazione del database.

Select

Permette di estrarre i dati, in modo mirato, dal database.

Sintassi del comando select

SELECT [ALL | DISTINCT | TOP] lista_elementi_selezione FROM lista_riferimenti_tabella
[WHERE espressione_condizionale] [GROUP BY lista_colonne [HAVING Condizione]]
[ORDER BY lista_colonne];

dove:

lista_elementi_selezione è l'elenco dei campi da estrarre (separati tra loro con una virgola);
lista_riferimenti_tabella è l'elenco delle tabelle da cui estrarre i dati;
espressione_condizionale rappresenta l'elenco delle condizioni, ovvero dei requisiti che un campo deve rispettare per poter essere prelevato dall'interrogazione (le condizioni sono specificate mediante gli operatori di confronto, connettori logici e comparatori come *between*, *in*, *like*, *is null*);
lista_colonne è la colonna o le colonne che devono essere prese come riferimento per l'ordinamento dei dati in uscita.

```
SELECT DISTINCT cognome, nome,
citta_residenza
FROM utenti WHERE anni > = 18
ORDER BY cognome
```

XML cos'è?

Markup Language (XML) è un **meta-linguaggio** di markup, cioè un linguaggio che permette di definire altri linguaggi di markup. A differenza di HTML, XML **non ha tag** predefiniti e non serve per definire pagine Web né per programmare.

XML serve esclusivamente per definire altri linguaggi.

XML è un **insieme standard di regole sintattiche per modellare la struttura di documenti** e dati.

Questo insieme di regole, dette più propriamente specifiche, definiscono le modalità secondo cui è possibile crearsi un proprio linguaggio di markup.

Le specifiche ufficiali sono state definite dal W3C (Worl Wide Web Consortium) e sono consultabili a partire dall'indirizzo http://www.w3.org/XML.

Struttura del documento XML

Un documento XML è un file di testo che contiene una serie di **tag, attributi** e **testo** secondo regole sintattiche ben definite.

- è intrinsecamente caratterizzato da una **struttura gerarchica**: è composto da componenti denominati **elementi**. Ciascun elemento rappresenta un componente logico del documento e può contenere altri elementi (**sottoelementi**) o del testo.
- Gli elementi possono avere associate altre informazioni che ne descrivono le proprietà. Queste informazioni sono chiamate **attributi**.
- L'organizzazione degli elementi segue un ordine gerarchico o arboreo che prevede un elemento principale, chiamato **root element** o semplicemente root o radice che contiene l'insieme degli altri elementi del documento

Struttura gerarchica di un documento XML

La **struttura logica** di un documento XML dipende dalle scelte progettuali. Non esistono regole universali.

La struttura logica di un documento XML viene tradotta in una corrispondente **struttura fisica** composta di elementi sintattici chiamati **tag**. Questa struttura fisica viene implementata tramite un file di testo creato con un qualsiasi editor.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
- <csm_root>
  - <csm_info>
      <nome_normativa>OA</nome_normativa>
      <tipo>scheda di catalogo</tipo>
      <ver_numero>3.00</ver_numero>
      <data_crea>16122014</data_crea>
      <ente_schedatore>S67</ente_schedatore>
      <concessione/>
      <spedizione/>
      <note/>
      <numero_schede>00000001</numero_schede>
    </csm_info>
  - <schede>
    - <scheda>
      - <CD>
          <TSK>OA</TSK>
          <LIR>C</LIR>
        - <NCT>
            <NCTR>01</NCTR>
            <NCTN>00350814</NCTN>
          </NCT>
          <ESC>S67</ESC>
          <ECP>S67</ECP>
        </CD>
      - <OG>
        - <OGT>
            <OGTD>dipinto</OGTD>
            <OGTV>opera isolata</OGTV>
          </OGT>
        - <SGT>
            <SGTI>David con la testa di Golia</SGTI>
          </SGT>
        </OG>
```

Documento «ben formato»

- Ogni documento XML deve contenere un **unico elemento di massimo livello (root)** che contenga tutti gli altri elementi del documento. Le sole parti di XML che possono stare all'esterno di questo elemento sono i commenti e le direttive di elaborazione (per esempio, la dichiarazione della versione di XML)
- **Ogni elemento deve avere un tag di chiusura** o, se vuoti, possono prevedere la forma abbreviata (/>)
- Gli elementi devono essere opportunamente nidificati, cioè **i tag di chiusura devono seguire l'ordine inverso dei rispettivi tag di apertura**
- XML fa distinzione tra **maiuscole e minuscole,** (case sensitive) per cui i nomi dei tag e degli attributi devono coincidere nei tag di apertura e chiusura anche in relazione a questo aspetto
- I **valori degli attributi** devono sempre essere racchiusi tra **singoli o doppi apici**
- Un tag può iniziare con un lettera o un underscore (_) e può contenere lettere, numeri, il punto, l'underscore (_) o il trattino (-). Non sono ammessi spazi o altri caratteri

Documento «valido»

Un documento XML che rispetta le regole definite da una **grammatica** è detto valido per un particolare linguaggio.

E' necessario definire una **grammatica** per il linguaggio di markup che abbiamo ideato. Una grammatica è un insieme di regole che indica quali vocaboli (**elementi**) possono ess

Ma come si definisce una grammatica per descrivere un linguaggio di markup?

Attualmente due sono gli approcci più diffusi alla creazione di grammatiche per documenti XML: **Dtd** - Document Type Definition e **XML Schema**.ere utilizzati e con che **struttura** è possibile comporre frasi (documenti).

XML Schema: cos'è

un XML Schema è una descrizione formale di una grammatica per un linguaggio di markup basato su XML

un XML Schema utilizza la stessa sintassi XML per definire la grammatica di un linguaggio di markup.

uno XML Schema è un documento XML che descrive la grammatica di un linguaggio XML utilizzando un linguaggio di markup specifico.

XML Schema: a cosa serve?

- Definire gli elementi (tag) che possono apparire in un documento
- Definire gli attributi che possono apparire in un elemento
- Definire quali elementi devono essere inseriri in altri elementi (da ora in avanti li chiameremo child)
- Definire il numero degli elementi child
- Definire quando un elemento dev'essere vuoto o può contenere testo, elementi, oppure entrambi
- Definire il tipo per ogni elemento e per gli attributi (intero, stringa, ecc, ma anche tipi personalizzati)
- Definire i valori di default o fissi per elementi ed attributi

Struttura di XML Schema (xsd)

- uno XML Schema ha un root element che contiene tutte le regole di definizione della grammatica rappresentato dal tag **<xs:schema>**.
- XML Schema prevede il tag **<xs:element>** per la definizione degli elementi utilizzabili in un documento XML, specificando nell'attributo name il nome del relativo tag. All'interno di ciascun tag <xs:element> possiamo indicare il tipo di dato dell'elemento e possiamo definire gli eventuali attributi.
- XML Schema prevede elementi **semplici** o **complessi**
  - **gli elementi semplici** non possono contenere altri elementi e avere attributi. Possono contenere solo testo.
  - **gli elementi complessi** possono contenere testo, altri elementi e attributi in qualsiasi combinazione.

Elementi semplici

tipo di dato semplice elemento che non può contenere altri elementi e non prevede attributi.

Si possono usare tipi di dato semplici predefiniti oppure è possibile personalizzarli.

Sono previsti numerosi **tipi di dato predefiniti**

```
<xs:element name="xxx" type="yyy" />
```

dove al posto di xxx è sufficiente scrivere il nome che si desidera usare per l'elemento mentre yyy va sostituito con il tipo dell'elemento (che può far parte dei tipi base o di quelli che abbiamo definito noi).

Per gli elementi è anche possibile definire un valore di default (sarà assegnato quando nessun altro valore è specificato) o un valore fisso (che sarà sempre automaticamente assegnato all'elemento).

xs:string	Stringa di caratteri
xs:integer	Numero intero
xs:decimal	Numero decimale
xs:boolean	Valore booleano
xs:date	Data
xs:time	Ora
xs:uriReference	URL

Elementi complessi

Definire un attributo per un elemento complesso è altrettanto semplice.

```
<xs:attribute name="xxx" type="yyy" />
```

dove xxx va sostituito con il nome dell'attributo e yyy con il suo tipo.

(Da notare che un attributo è considerato un elemento di tipo semplice e pertanto non può contenere altri elementi e avere attributi. Il fatto che l'attributo sia considerato un tipo semplice spiega anche perchè un elemento che lo contiene dev'essere necessariamente di tipo complesso.)

Un tipo complesso può essere infatti di molti tipi diversi:

- può essere un elemento vuoto
- può contenere solo elementi
- può contenere solo testo
- può contenere sia elementi che testo

- **Gli indicatori** principali sono quelli di ordine e di presenza.
- **All** specifica di default che gli elementi child possono apparire in qualsiasi ordine e che ogni elemento child deve apparire una ed una sola volta.
- **Choice** specifica che gli elementi in esso contenuti possono apparire in alternativa l'uno all'altro.
- **sequence** specifica che gli elementi child devono apparire nell'ordine specifico (quello con cui sono dichiarati al suo interno).
- **maxOccurs** specifica quante volte un elemento può ripetersi
- **minOccurs** specifica quanti elementi di quel tipo devono ripetersi perchè il documento sia considerato valido

Attributi degli elementi

La definizione degli attributi è basata sull'uso del tag <*xs:attribute*>, come nel seguente esempio:

<xs:attribute **name**="titolo" **type**="xs:string" **use**="required" />

- L'attributo **use c**onsente di specificare alcune caratteristiche come la presenza obbligatoria (required) o un valore predefinito (default) in combinazione con l'attributo **value.** Ad esempio, la seguente definizione indica un attributo il cui valore di predefinito è test:

<xs:attribute name="titolo" type="xs:string" use="default" value="test" />

(se non si specifica esplicitamente l'obbligatorietà dell'attributo, esso è considerato opzionale)

.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="scheda" id="tag_normativa">
<xs:complexType>
<xs:sequence>
<xs:element name="CD" id="paragrafo_CD" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:sequence>
<xs:element name="TSK" id="camposemplice_OA_CD_TSK" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="alias" type="xs:string" use="optional" fixed="Tipo Scheda"/>
<xs:attribute name="len" type="xs:string" use="optional" fixed="0,4"/>
<xs:attribute name="node_visibility" type="xs:string" use="optional" fixed="1"/>
<xs:attribute name="node_linkMandatory" type="xs:string" use="optional" fixed="true"/>
<xs:attribute name="binding_thesId" type="xs:string" use="optional" fixed="VC_TSK_OA"/>
<xs:attribute name="binding_levelExpr" type="xs:string" use="optional" fixed="$1"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
<xs:element name="LIR" id="camposemplice_OA_CD_LIR" minOccurs="1" maxOccurs="1">
<xs:complexType>
<xs:simpleContent>
<xs:extension base="xs:string">
<xs:attribute name="alias" type="xs:string" use="optional" fixed="Livello ricerca"/>
<xs:attribute name="len" type="xs:string" use="optional" fixed="0,5"/>
<xs:attribute name="node_visibility" type="xs:string" use="optional" fixed="1"/>
<xs:attribute name="node_linkMandatory" type="xs:string" use="optional" fixed="true"/>
<xs:attribute name="binding_thesId" type="xs:string" use="optional" fixed="VC_LIR"/>
<xs:attribute name="binding_levelExpr" type="xs:string" use="optional" fixed="$1"/>
</xs:extension>
</xs:simpleContent>
</xs:complexType>
</xs:element>
```