

# CAP7\_METODOLOGIE\_MODELLI

## Table of contents

- [Preoccuparci della modellazione](#)
- [Progettazione di DB](#)
  - [Progettazione](#)
    - [Modello dei dati](#)
      - [Modello Entity-Relationship \(E-R\)](#)
        - [Entità](#)
        - [Relationship](#)
        - [Attributo](#)
        - [Cardinalità](#)
        - [Identificatori](#)
        - [Generalizzazioni](#)

## Preoccuparci della modellazione

L'idea è che in una fase iniziale di progetto, lo schema logico dei dati sia troppo dettagliato: è quello che diamo in mani a chi deve implementare il DB quando noi stiamo ancora decidendo se costruire o meno la casa. Serve del tempo prima che la bozza si avvicini alla soluzione.

## Progettazione di DB

#modelling

#modello-concettuale

#schema\_e-r

Ci concentriamo su 2 aspetti:

Raccolta e analisi dei requisiti

Progettazione

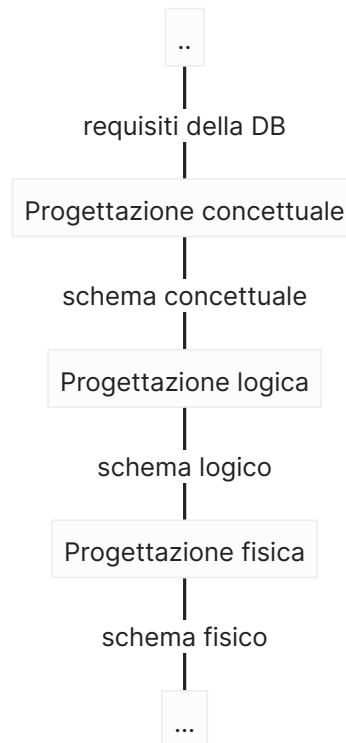
La complicità arriva quando *human-say-how* entra in gioco. A volte le persone sanno fare le cose ma quando le raccontano fanno molti errori. Gestire tutti i problemi che un programma potrebbe avere, è necessario per aspettarci un buon funzionamento.

In questo corso ci aspettiamo che la "Raccolta e analisi dei requisiti" sia già stata fatta, di solito da chi ha molta esperienza siccome capiscono presto quali sono le necessità dell'utente.

Seguire una *metodologia di progetto*:

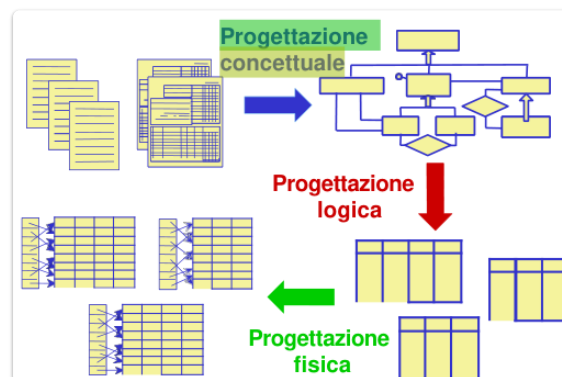
- articolazione delle fasi;
- criteri di scelta  
la soluzione dove è? proviamo a capire i criteri per arrivarci se ci accorgiamo che la strada è sbagliata;
- modelli di rappresentazione
- generalità e facilità d'uso  
non deve essere troppo complicato farla

# Progettazione



I risultati delle varie fasi di progettazione sono schemi di *modelli di dati*; usiamo la rappresentazione grafica. Capiremo come fare, in modo dettagliato, lo *schema concettuale*, con un linguaggio standard che sarà quello del libro di testo.

## Modello dei dati



Sarebbe l'insieme di costrutti che consentono di definire i dati e come si comportano. Questi costrutti sono le *tabelle*, gli *attributi*, i *vincoli*, tutto quello che compone una DB.

- **schema**  
come sono strutturati i miei dati, aspetto intensionale;
- **istanza**  
i dati attuali, variano molto rapidamente nel tempo, aspetto estensionale.

I modelli sono:

- **logici**  
per l'organizzazione dei dati, utilizzati dai programmi, indipendenti dalle strutture fisiche;

- **concettuali**




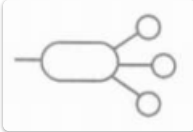

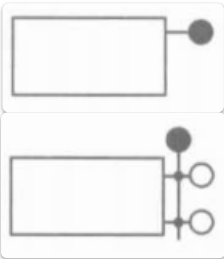
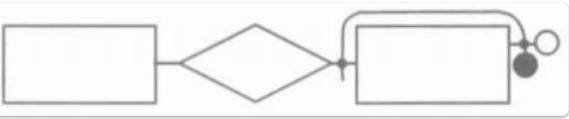


non possono essere utilizzati direttamente dai programmi

uno dei primi proposti è quello **Entità-Relazione**, il più noto e utilizzato.

## Modello Entity-Relationship (E-R)

### ⚠ Nomenclatura degli schemi E-R

Nelle immagini che seguiranno, i nomi delle entità saranno al singolare, ma nella pratica sarà da usarsi la nomenclatura al plurale.

Costrutti	Rappresentazione grafica
Entita'	
Relazione	
Attributo	
Attributo composto	
Cardinalita'	
Identificatore interno	
Identificatore esterno	
Generalizzazione	
Sottoinsieme	

## Entità

Un qualcosa che ha caratteristica comune, del mondo reale, materiale o immateriale. Nella progettazione stiamo decidendo quali sono gli aspetti della realtà che ci interessano, il resto non conta. Vogliamo rappresentare che quell'identità esiste senza associazioni.

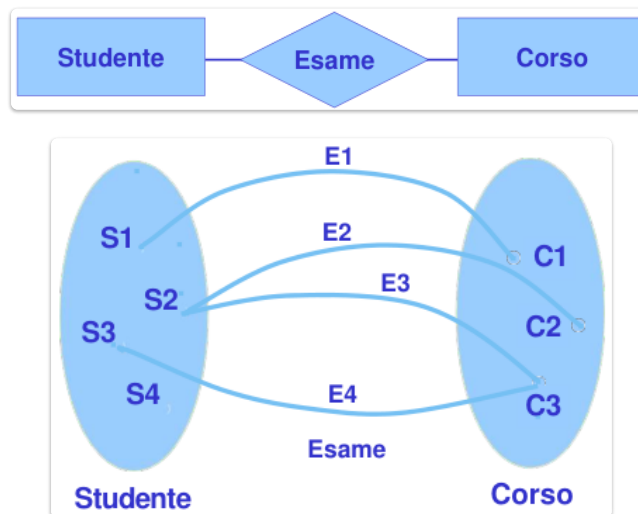
- si parla di **occorrenza** quando ci stiamo riferendo all'entità del mondo esterno, come ad esempio una persona
- ogni identità ha **nome** che la identifica e devono essere *espressivi* e con *opportune convenzioni*



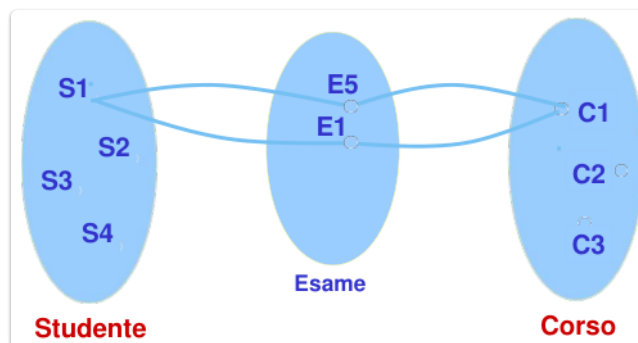
## Relationship

Legami logici tra 2 o più insiemi di Entità nell'applicazione d'interesse.

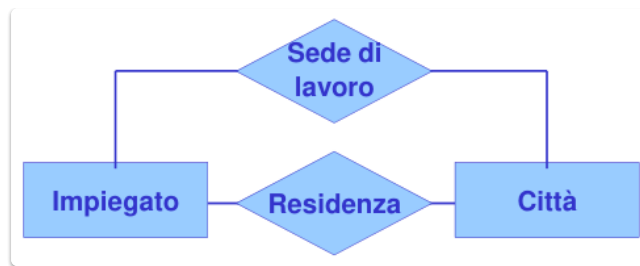
- **nome** univoco non ripetibile nello schema;
- usando convenzioni quali **sostantivi** invece che verbi;



Non possono esistere due occorrenze di `Esame` per uno solo `Studente - Corso`. Possiamo però *promuovere* per permetterlo.



La stessa copia di entità può essere collegata da più relazioni, tenendo sempre conto che una volta creata, una tupla non può essere ripetuta.

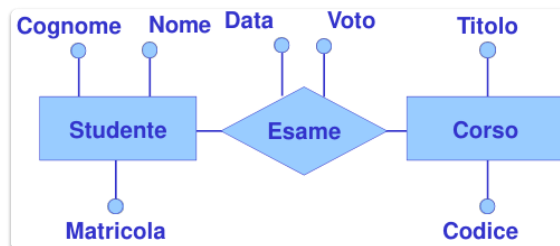


Associazioni ternarie esistono ma sono rare e spesso complicano la situazione.

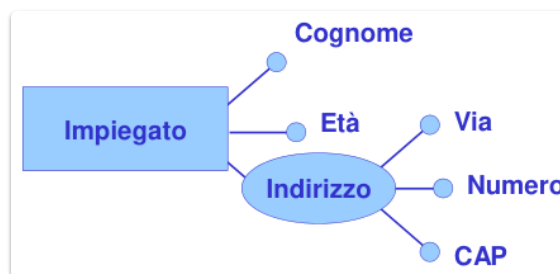
## Attributo

Sarebbe una proprietà elementare dell'entità su cui stiamo ragionando.

Associa a ogni occorrenza di entità o relationship un valore appartenente a un insieme detto *dominio* dell'attributo.



Possono essere composti, nel senso che una medesima entità o relationship presenta affinità nel loro significato/uso.



## Cardinalità

### ✍ Nomenclatura delle associazioni 1 a molti

Puo' essere omessa. Non e' necessario dargli un nome in quanto il concetto e' esplicitato dal nome delle entita' coinvolte.

### ✍ Nomenclatura delle associazioni molti a molti

Deve sempre essere precisato un nome.

Coppia di valori associati a ogni entità che partecipa a una relationship.

Specificano il numero minimo e massimo di occorrenze delle relationship cui ciascuna occorrenza di un'entità può partecipare.

- 0 per cardinalità minima, "partecipazione *opzionale*";
- 1 per cardinalità minima, "partecipazione *obbligatoria*";

- $N$  per massima, nessun limite.



I **tipi di relationship** sono:

- *uno a uno* (molto rare);
- *uno a molti*;



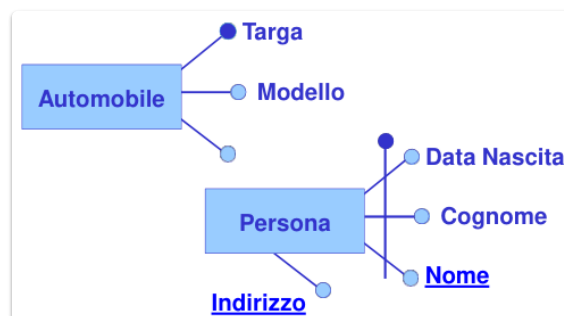
- *molti a molti*.



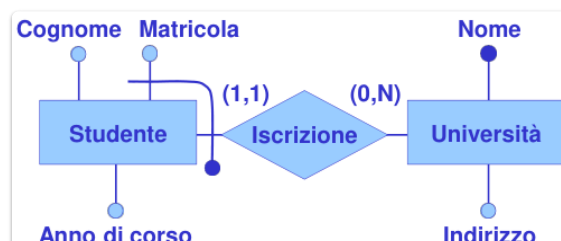
## Identificatori

Per identificare univocamente, le occorrenze di un'entità.

- **Interno**  
attributi dell'entità



- **esterno**  
attributi + entità esterne attraverso relationship



## Generalizzazioni

Si dice **totale** se ogni occorrenza dell'entità genitore è una occorrenza di almeno una delle entità figlie, altrimenti è **parziale**.

### Generalizzazione totale esclusiva

Tutte le Persone sono Uomo e Donna, una persona e' o Uomo o Donna.



### Generalizzazione parziale esclusiva

Ogni Professionista ha una sola professione principale, ma esistono anche altre professioni.

