Basi Di Dati e di conoscenza

Progettazione Concettuale – Entità relazioni

Modello dei Dati

- insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica
- componente fondamentale: meccanismi di strutturazione (o costruttori di tipo)
- come nei linguaggi di programmazione esistono meccanismi che permettono di definire tipi di dati, così ogni modello dei dati prevede alcuni costruttori
- ad esempio, il modello relazionale prevede il costruttore relazione, che permette di definire insiemi di record omogenei

Modelli dei dati

- modelli logici: utilizzati nei DBMS esistenti per l'organizzazione dei dati
  - utilizzati dai programmi
  - indipendenti dalle strutture fisiche

  esempi: relazionale, reticolare, gerarchico, a oggetti
- modelli concettuali: permettono di rappresentare i dati in modo indipendente da ogni sistema e dal modello logico su cui è basato
  - cercano di descrivere i concetti del mondo reale
  - sono utilizzati nelle fasi preliminari di progettazione

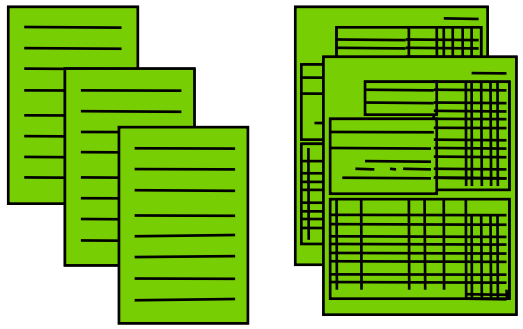
  il più noto è il modello **Entità-Relazione.**
- Sono stati sviluppati modelli più moderni:
  - **Modello Entità Relazioni Esteso –EER** (consente l'utilizzo di costrutti più potenti e.g. ereditarietà)
  - **UML**: linguaggio unificato per la progettazione dei dati e delle funzioni.

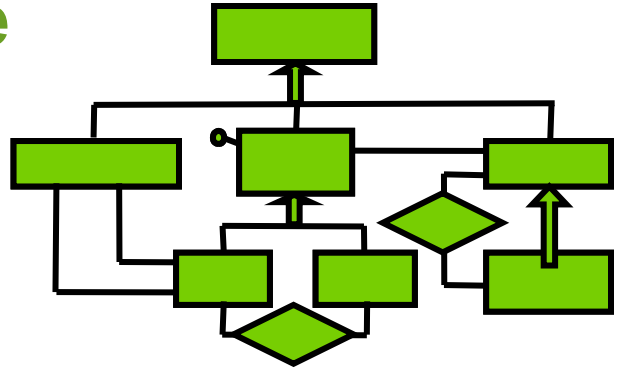
Progettazione di Basi di Dati

- Progettare una base di dati significa definirne **struttura**, **caratteristiche** e **contenuto**.
- Prevede l'uso di opportune **metodologie**. In base al grado di **astrazione**, la progettazione abbiamo:

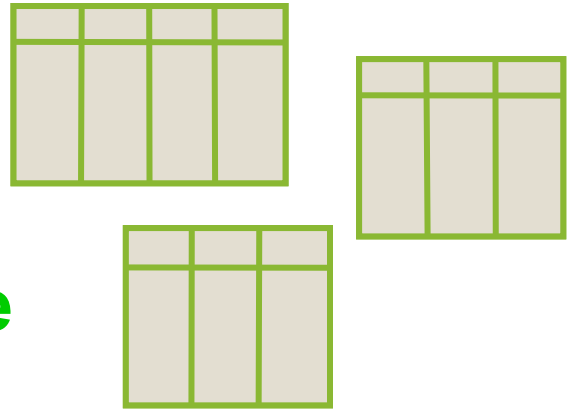
1. **Modello concettuale**: rappresenta la realtà dei dati e le relazioni tra essi attraverso uno schema
2. **Modello logico**: descrive il modo attraverso il quale i dati sono organizzati negli archivi del calcolatore
3. **Modello fisico:** descrive come i dati sono registrati nelle memorie di massa

Progettazione concettuale

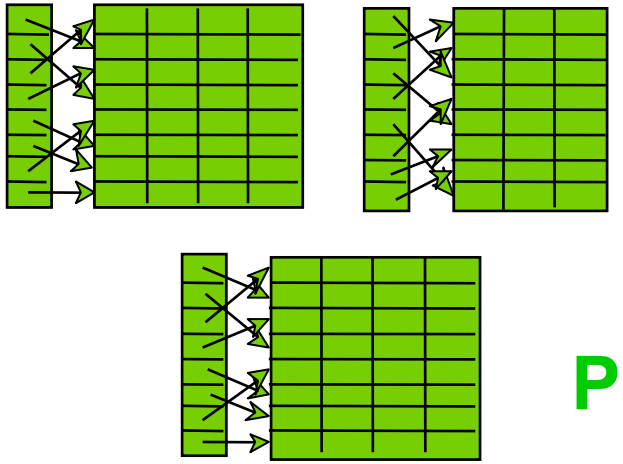




Progettazione logica



Progettazione fisica



Modello concettuale

- Rappresentazione **astratta** della realtà:
  - definire un insieme di dati presenti in natura e che rappresentano la natura stessa delle informazioni che si vogliono archiviare
  - **non esistono regole** prefissate per l'individuazione dei dati e per la loro selezione

Progettazione della base di dati

1. Cosa c'è? (Oggetti)
2. Come si collegano o si parlano?
3. Quanti tra loro?
4. Cosa identifica gli "oggetti"?
5. Quali informazioni utili non principali?

Modello Entità Relazioni ER

- **ER=Entity/Relationship**

- Il modello **entità relazione** è uno strumento per analizzare le caratteristiche di una realtà in modo indipendente dagli eventi che in essa accadono, cioè per costruire un modello concettuale dei dati indipendente dalle applicazioni.

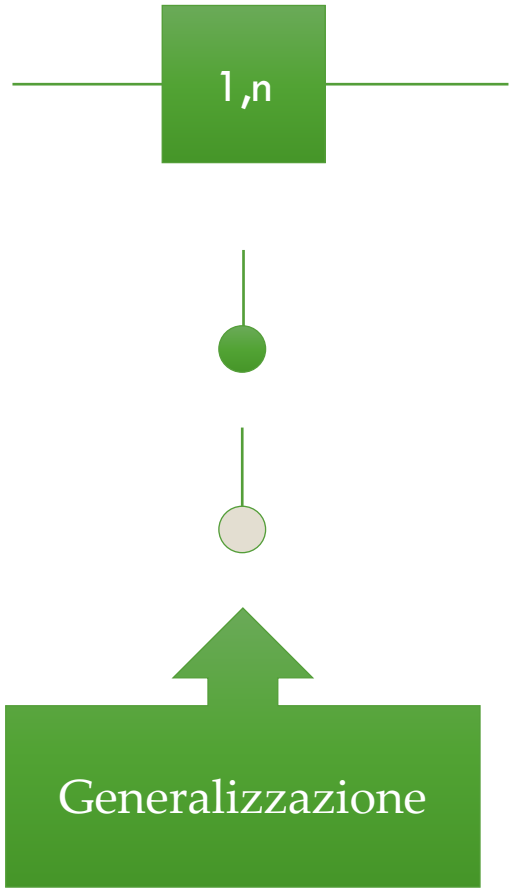
Modello Entità Relazioni ER

insieme di oggetti simili, dati dello stesso tipo o con caratteristiche simili, raccolti insieme.

Statici e Dinamici

Collegamento logico tra due o più entità.

Modello Entità Relazioni ER



Cardinalità: assegnazione di dimensionamento tra entità. Numero *min* e *max* di possibili collegamenti tra due entità tramite una relazione

Chiave: Campo o campi indentificativi di una entità o relazione

Attributi: Campi informativi e non identificativi di una entità o relazione

legame logico tra una entità più **generale (padre)** e le entità **figlie**.

Modello ER - Entità

Entità

- E' un oggetto, concreto o astratto, che ha un significato anche quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare

Esempio

- le persone, un modello di automobile, i movimenti contabili, le prove sostenute da uno studente
- gli studenti sono classificabili nel tipo entità Studente (Il singolo studente rappresenta un'istanza del tipo entità Studente)
- i diversi modelli di automobile sono classificabili nel tipo entità Automobile

Modello ER - Entità

Entità

- E' un oggetto, concreto o astratto, che ha un significato anche quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare

Esempio

Studente

Automobile

Persona

Modello ER - Relazione

Relazione

- La **relazione** (**relationship**) è un legame che stabilisce un'interazione tra le entità

Esempio

- tra entità **Persona** e l'entità **Automobile** esiste l'associazione **Possedere** (tra Automobile e Persona esiste l'associazione Essere posseduta che è l'inverso) che può essere descritta nel linguaggio naturale secondo due versi
  - una persona può possedere una o più automobili
  - un'automobile deve essere posseduta da una persona

Modello ER - Relazione

Relazione

- La **relazione** (**relationship**) è un legame che stabilisce un'interazione tra le entità

Esempio

- Si rappresenta un solo verso della relazione

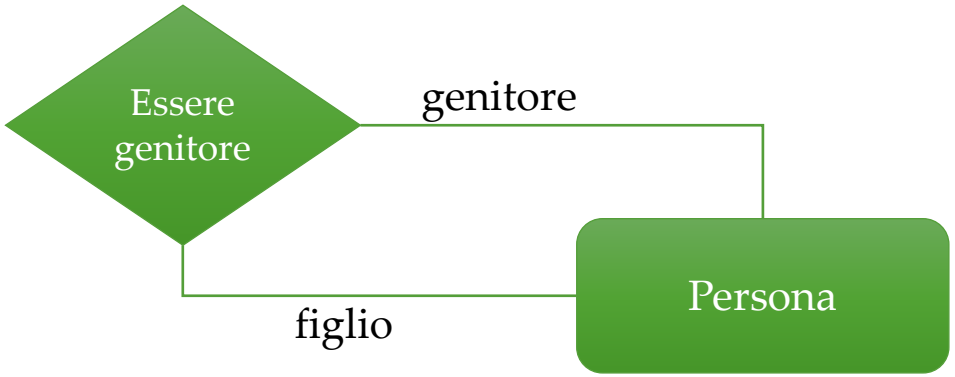
Modello ER - Relazione

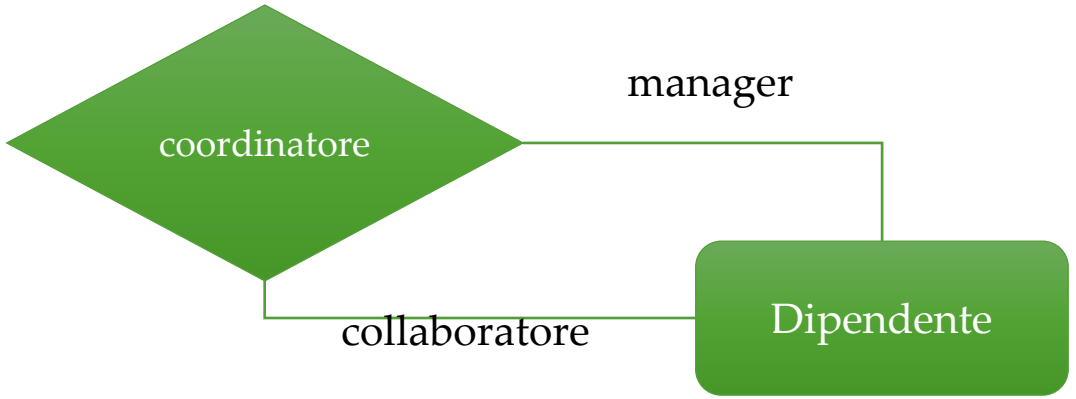
Relazione

- Le relazioni tra un'entità e se stessa si dicono **ricorsive o isA**

Esempio

- Esempio di relazione ricorsiva sull'entità Persona è l'associazione **Essere genitore** nella quale **Persona** partecipa con il ruolo di **Genitore** e di **Figlio**





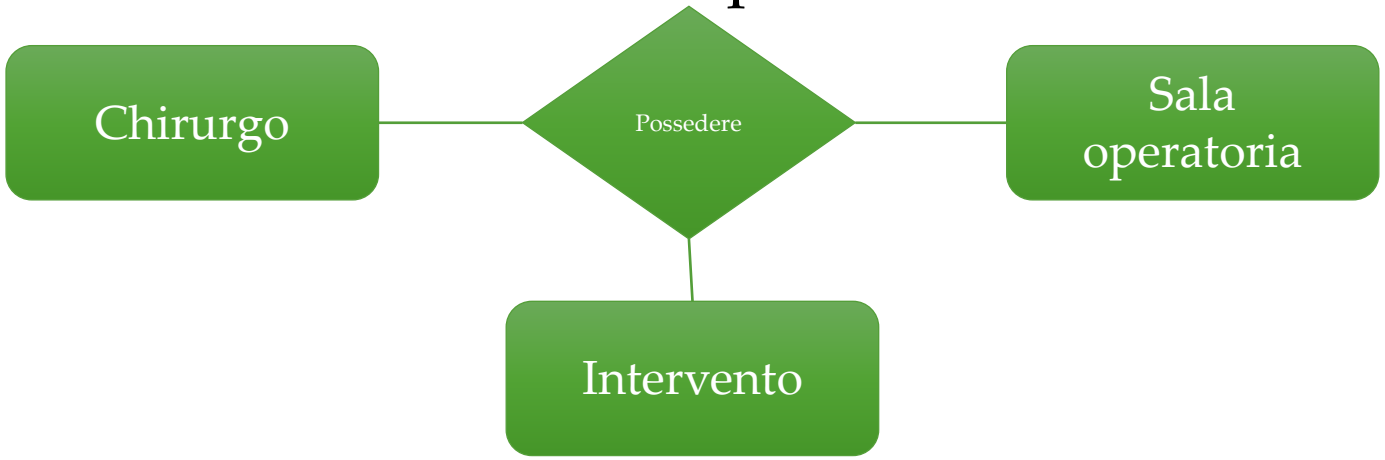
Modello ER - Relazione

Relazione

- Le relazioni tra tre entità si dicono **ternarie**

Esempio

- Una relazione ternaria si scompone in diverse relazioni binarie



Modello ER – Attributi e Chiavi

Attributi

Le proprietà delle entità e delle relazioni sono descritte tramite gli **attributi**. Alcune caratteristiche che descrivono il **Dominio**

- **Formato**: tipi di valore che assume (carattere, numerico, data/ora, …)
- **Dimensione**: quantità max di caratteri o cifre inseribili
- **Opzionalità**: possibilità di non essere sempre valorizzato

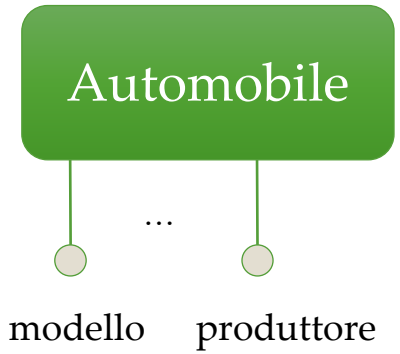
Esempio

Automobile (<u>numero telaio</u>, modello, produttore, cilindrata, prezzo)

Modello ER – Attributi e Chiavi

Esempio

Automobile (<u>numero telaio</u>, modello, produttore, cilindrata, prezzo)



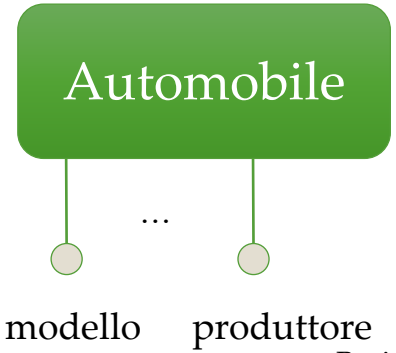
Modello ER – Attributi e Chiavi

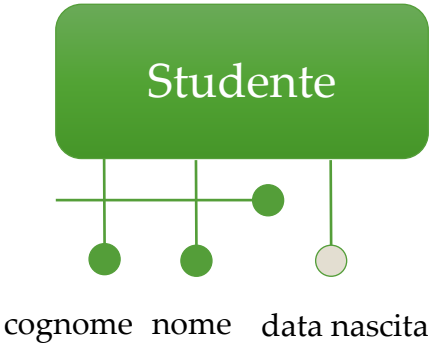
Attributi

La **chiave primaria** è un attributo o un insieme di attributi che identificano univocamente un'istanza dell'entità

Esempio

- **Automobile** (<u>numero telaio</u>, modello, produttore, cilindrata, prezzo)
- **Studente (**<u>cognome, nome</u>, data nascita)



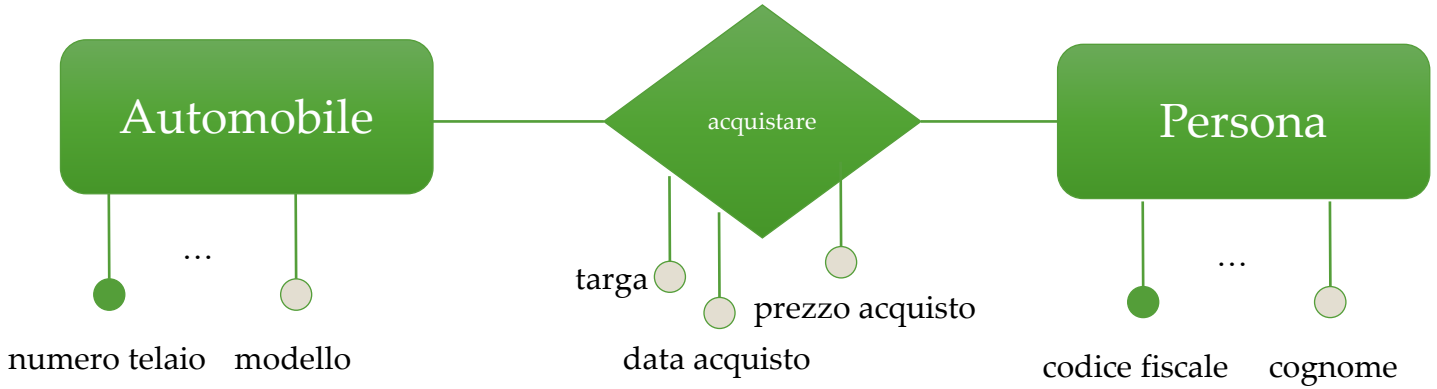


Modello ER – Attributi e Chiavi

Attributi

Anche le relazioni possono avere attributi

Esempio

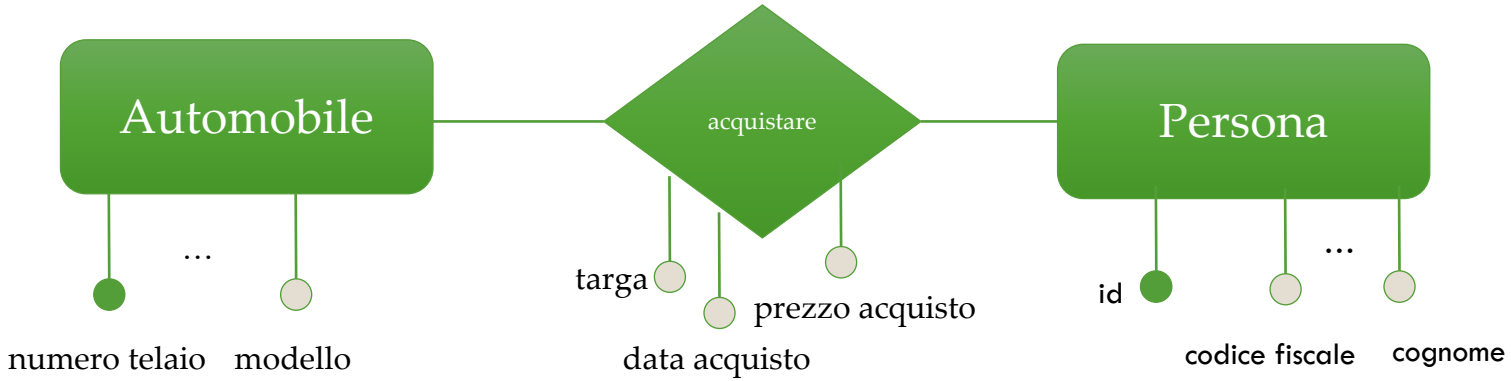


Modello ER – Attributi e Chiavi

Attributi

Spesso, anche in presenza di chiavi palesi, si utilizza un numero progressivo come chiave primaria ovvero una chiave artificiale

Esempio

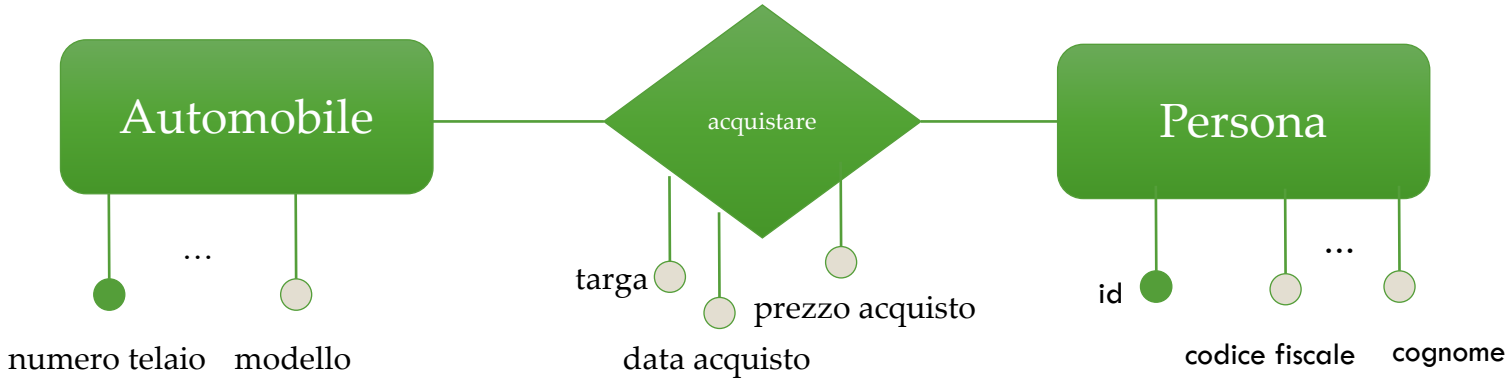


Modello ER – Attributi e Chiavi

Attributi

Una chiave artificiale è formata da un attributo privo di significato proprio. Di solito consiste in un contatore che si autoincrementa ad ogni istanza che si aggiunge

Esempio



Modello ER - Proprietà

Proprietà

- Ogni entità deve avere una **chiave primaria**
- L'attributo chiave primaria **non può essere opzionale**
- La chiave primaria **non può avere valori ripetuti**

Modello ER - Proprietà

Dominio

- **Tipo di dato**: intero, decimale, carattere, data, …
- **Lunghezza:** numero di cifre o caratteri per rappresentare il valore dell'attributo
- **Intervallo:** limite superiore e inferiore dei valori
- **Vincoli:** restrizioni sui valori ammessi
- **Supporto del valore NULL**, quando non è assegnato nessun valore
- Valore di **default**

Per le chiavi primarie:

- Il valore deve essere unico e i NULL non son ammessi

Per le chiavi esterne:

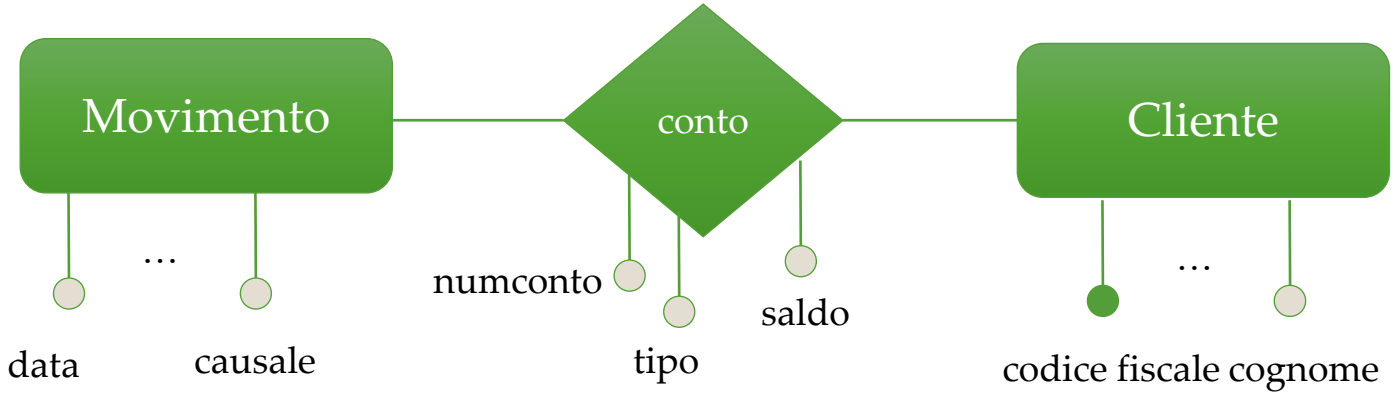
- Iltipo di dato, la lunghezza e il formato della chiave esterna devono essere uguali a quelli della corrispondente chiave primaria

Modello ER - Entità

Entità deboli e forti

- Entità che non hanno una chiave primaria e devono essere associate ad un'altra entità per essere completamente significative prendono il nome di **entità deboli**

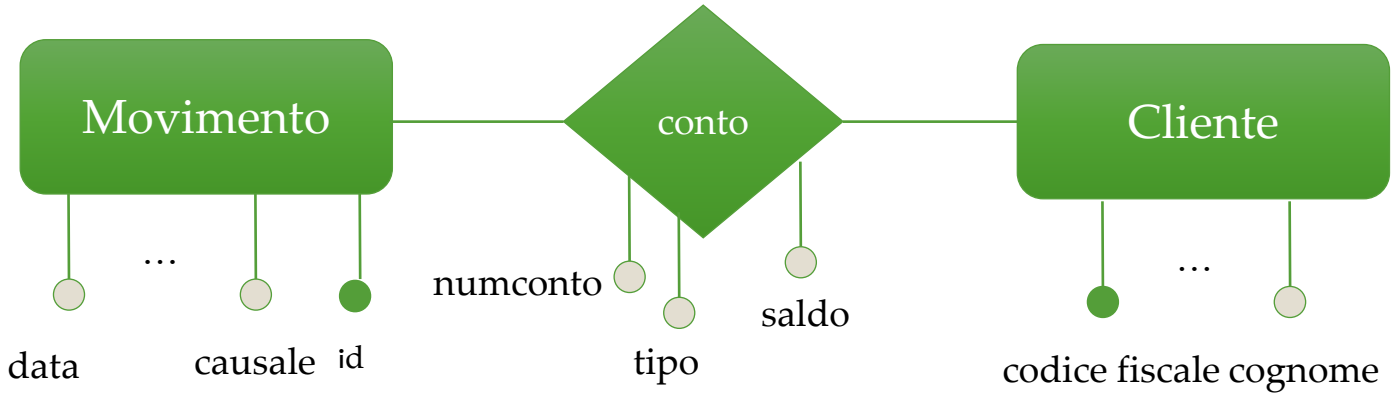
Esempio



Modello ER - Entità

Esempio: Entità deboli e forti

- Movimento ha senso solo in relazione a Conto
- Movimento è un entità debole
- Cliente e Conto sono entità forti
- Per evitare entità deboli, si aggiunge un numero progressivo come attributo



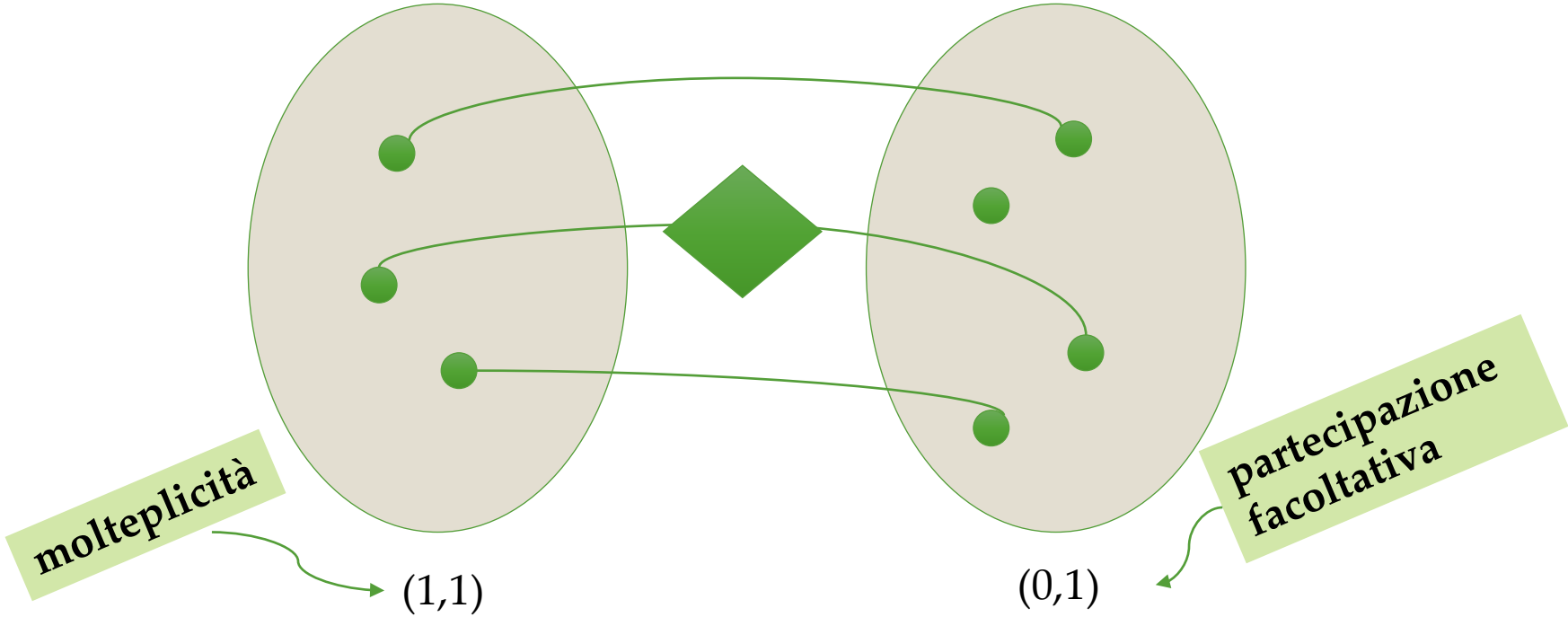
Modello ER - Molteplicità

La **molteplicità** di una relazione è il numero di possibili istanze di un'entità che sono messe in corrispondenza con un'istanza dell'altra entità

- I valori **min** e **max** assunti dalla molteplicità sono rappresentati con: **(1,1), (1,N), (0,1), (0,N)**
- Al valore **min** è associato il concetto di partecipazione facoltativa **(0)** o obbligatoria **(1)**
- Il valore **massimo** definisce la cardinalità della partecipazione all'associazione e assume i due valori **1** e N.
- In relazione alla cardinalità si parla di **associazioni**:
  - **Uno a uno 1:1**
  - **Uno a molti 1:N (Molti a uno N:1)**
  - **Molti a molti N:N**

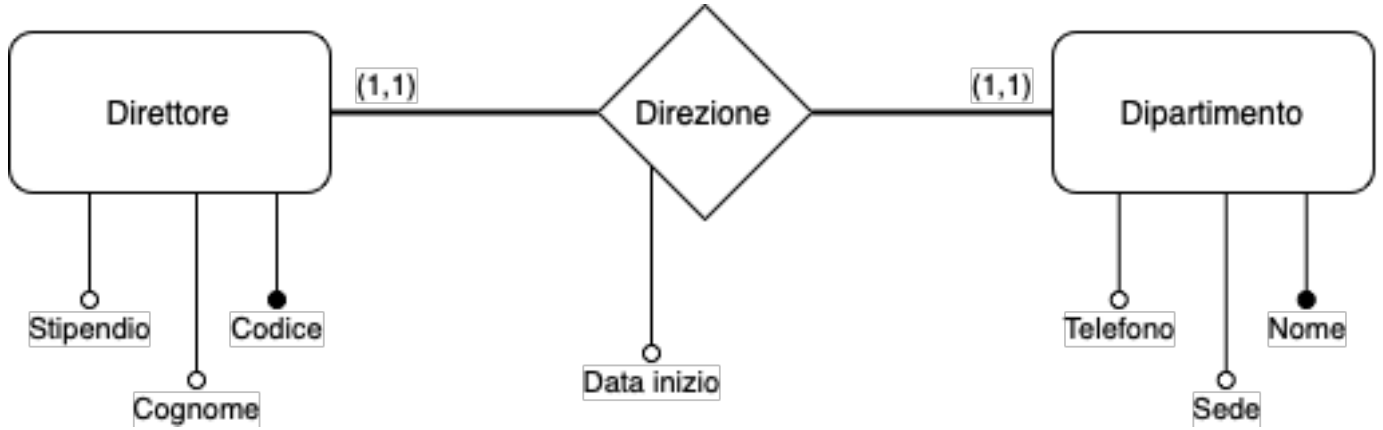
Modello ER – Associazione 1:1

Un'associazione tra **E1** ed **E2** si dice **uno a uno (1:1)** quando a ogni istanza di **E1** corrisponde una sola istanza di **E2** e a ogni istanza di **E2** corrisponde una sola istanza di **E1**.



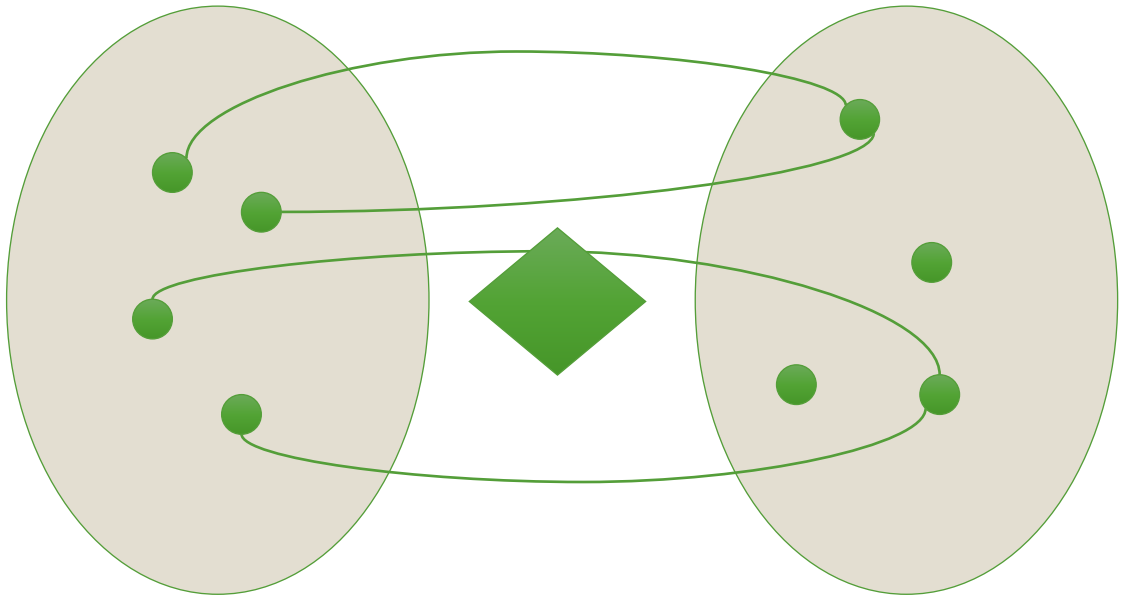
Modello ER – Associazione 1:1

Un'associazione tra **E1** ed **E2** si dice **uno a uno (1:1)** quando a ogni istanza di **E1** corrisponde una sola istanza di **E2** e a ogni istanza di **E2** corrisponde una sola istanza di **E1**.



Modello ER – Associazione 1:N (N:1)

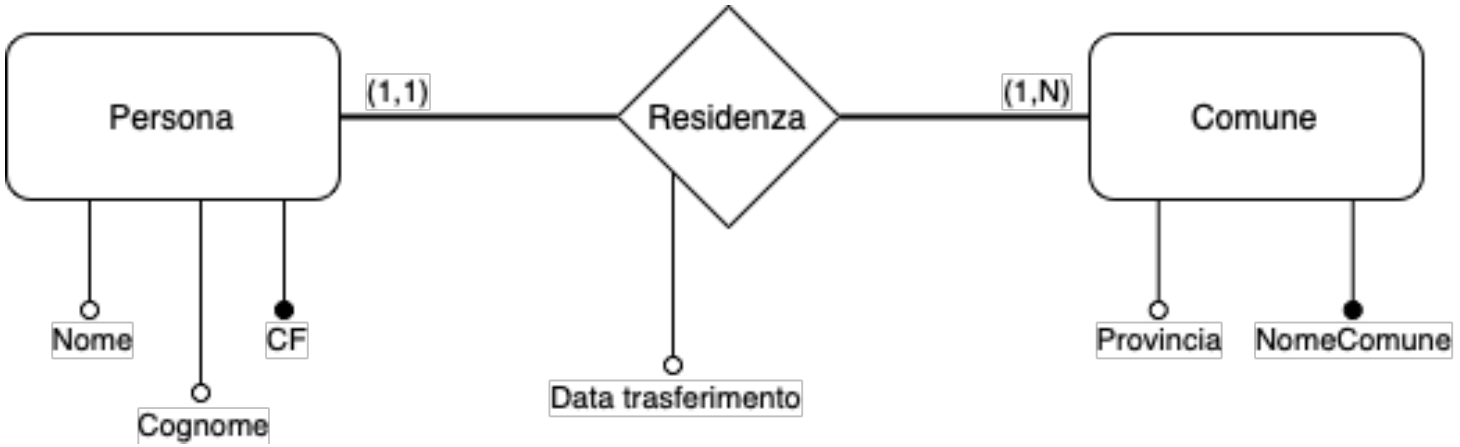
Un'associazione tra **E1** ed **E2** si dice **uno a molti (1:N)** quando a ogni istanza di **E1** corrispondono una o più istanze di **E2** e a ogni istanza di **E2** corrisponde una sola istanza di **E1**



(1,1) (0,N)

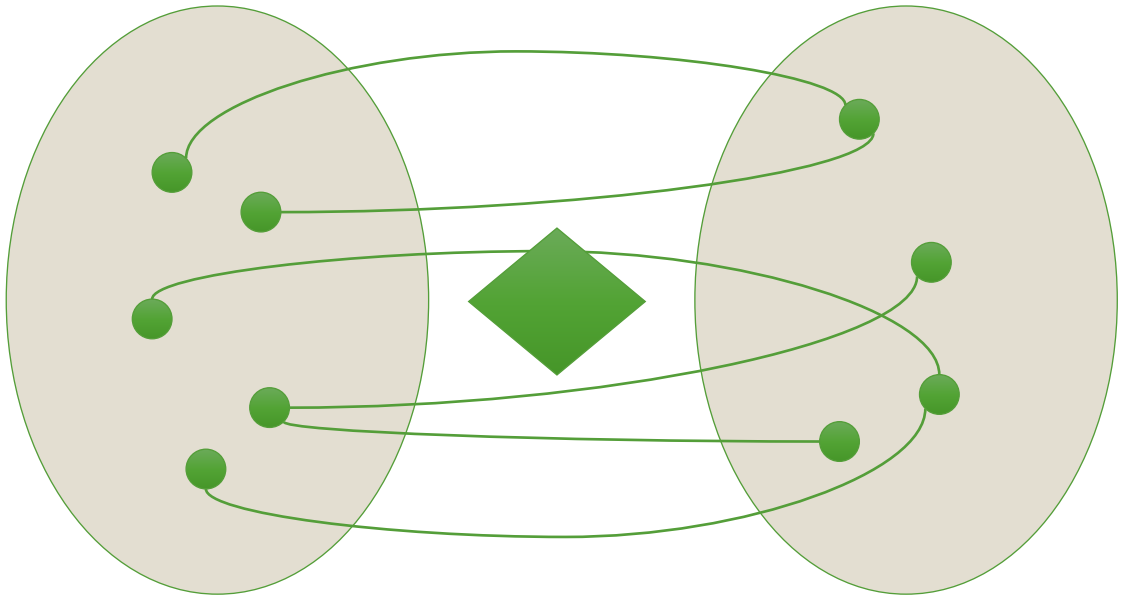
Modello ER – Associazione 1:N (N:1)

Un'associazione tra **E1** ed **E2** si dice **uno a molti (1:N)** quando a ogni istanza di **E1** corrispondono una o più istanze di **E2** e a ogni istanza di **E2** corrisponde una sola istanza di **E1**



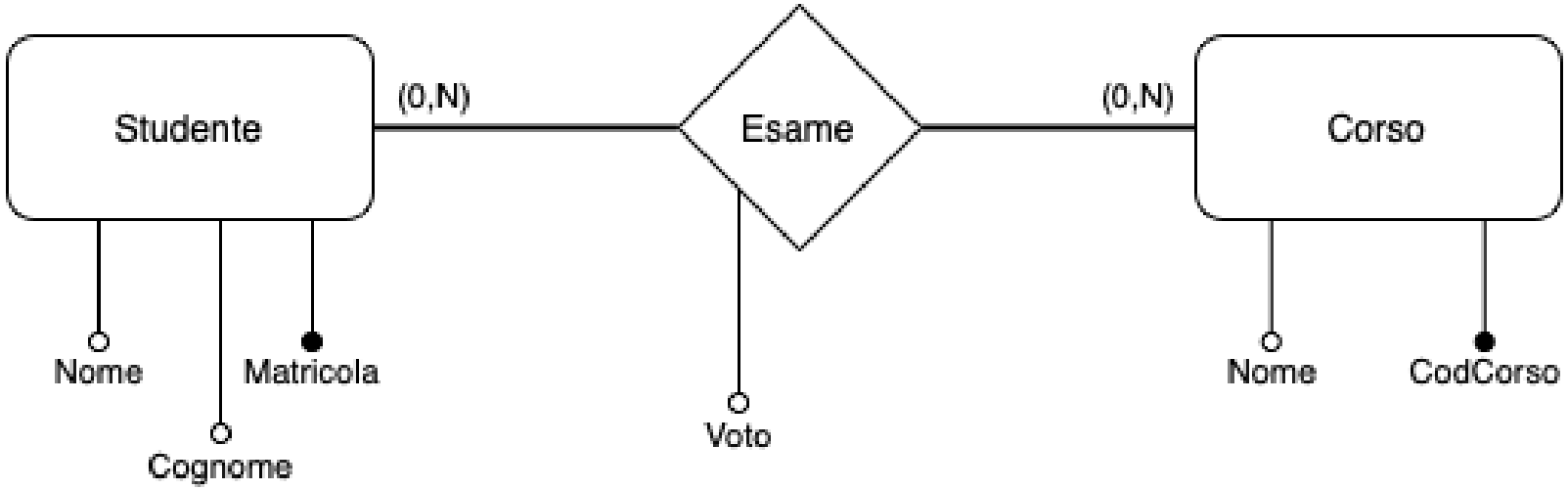
Modello ER – Associazione N:N

Un'associazione tra **E1** ed **E2** si dice **molti a molti (N:N)** quando a ogni istanza di **E1** corrispondono una o più istanze di **E2** e a ogni istanza di **E2** corrispondono una o più istanze di **E1**.



Modello ER – Associazione N:N

Un'associazione tra **E1** ed **E2** si dice **molti a molti (N:N)** quando a ogni istanza di **E1** corrispondono una o più istanze di **E2** e a ogni istanza di **E2** corrispondono una o più istanze di **E1**.



Modello ER - IsA

Può accadere che sussista **l'associazione IS-A** (o associazione di sottoinsieme) tra due entità,

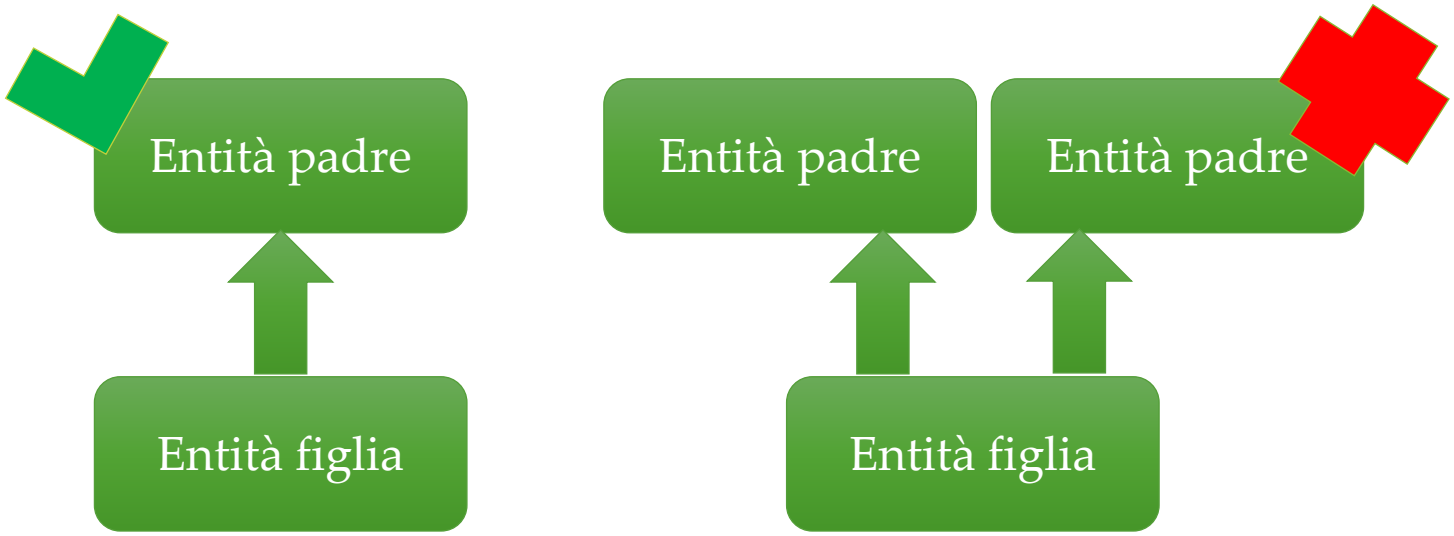
e cioè che ogni istanza di una sia anche istanza dell'altra.

La associazione IS-A nel modello ER si può definire tra due entità, che si dicono "entità padre" ed "entità figlia" (o sottoentità, cioè quella che rappresenta un sottoinsieme dell'entità padre)

Modello ER - IsA

L'associazione IS-A si rappresenta nel diagramma dello schema concettuale mediante una freccia dalla sottoentità alla entità padre.

Vige la regola che un'entità può avere al massimo una entità padre. In altre parole, il modello ER non ammette ereditarietà multipla



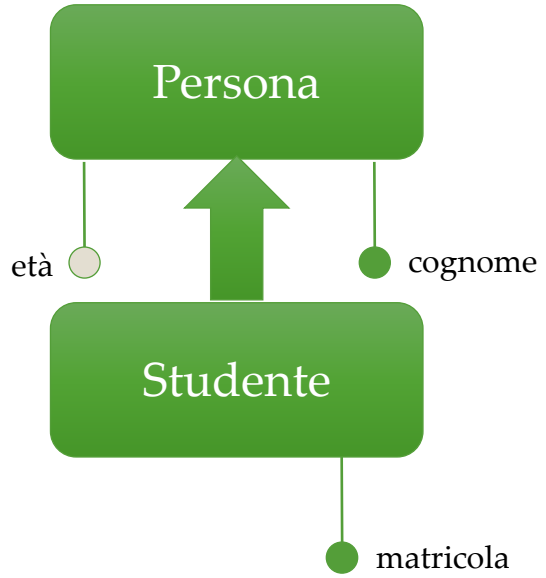
Modello ER - Ereditarietà

Principio di ereditarietà: ogni proprietà dell'entità padre è anche una proprietà della sottoentità, e non si riporta esplicitamente nel diagramma.

L'entità figlia può avere ovviamente ulteriori proprietà.

Esempio

cognome ed **età** ereditati da **Persona**

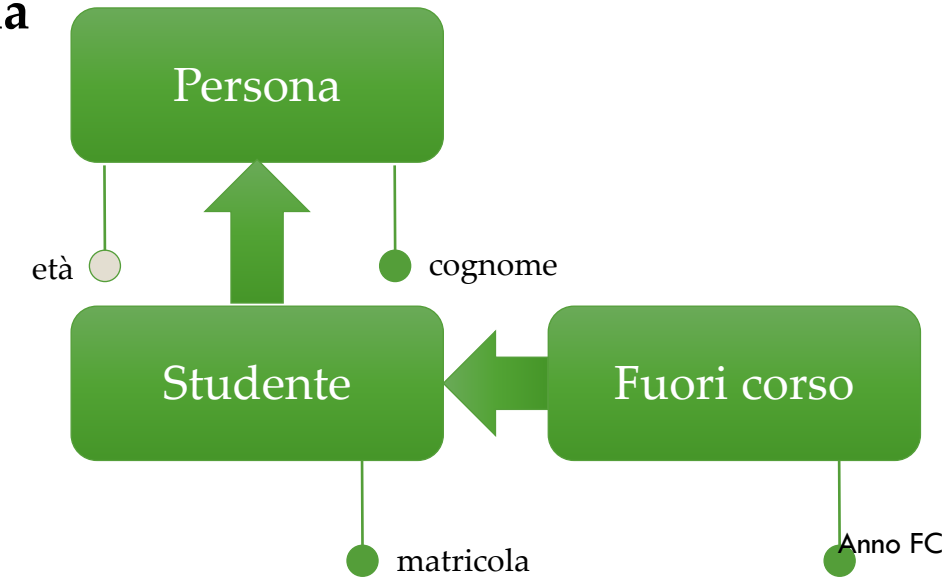


Modello ER - Ereditarietà

Principio di ereditarietà: l'associazione IS-A si eredita, pertanto IS-A è transitiva

Esempio

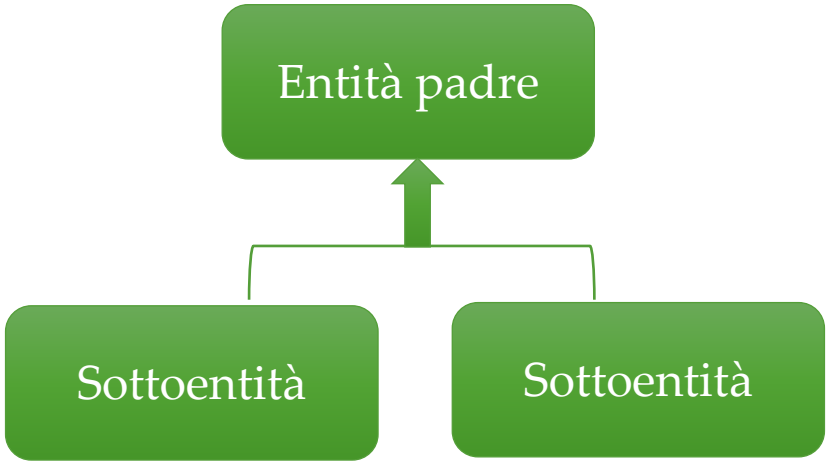
ogni istanza di **Studente** è un'istanza di **Persona**, ogni istanza di **Fuori corso** è un'istanza di **Studente** → ogni istanza di **Fuori corso** è un'istanza di **Persona**



Modello ER - Generalizzazione

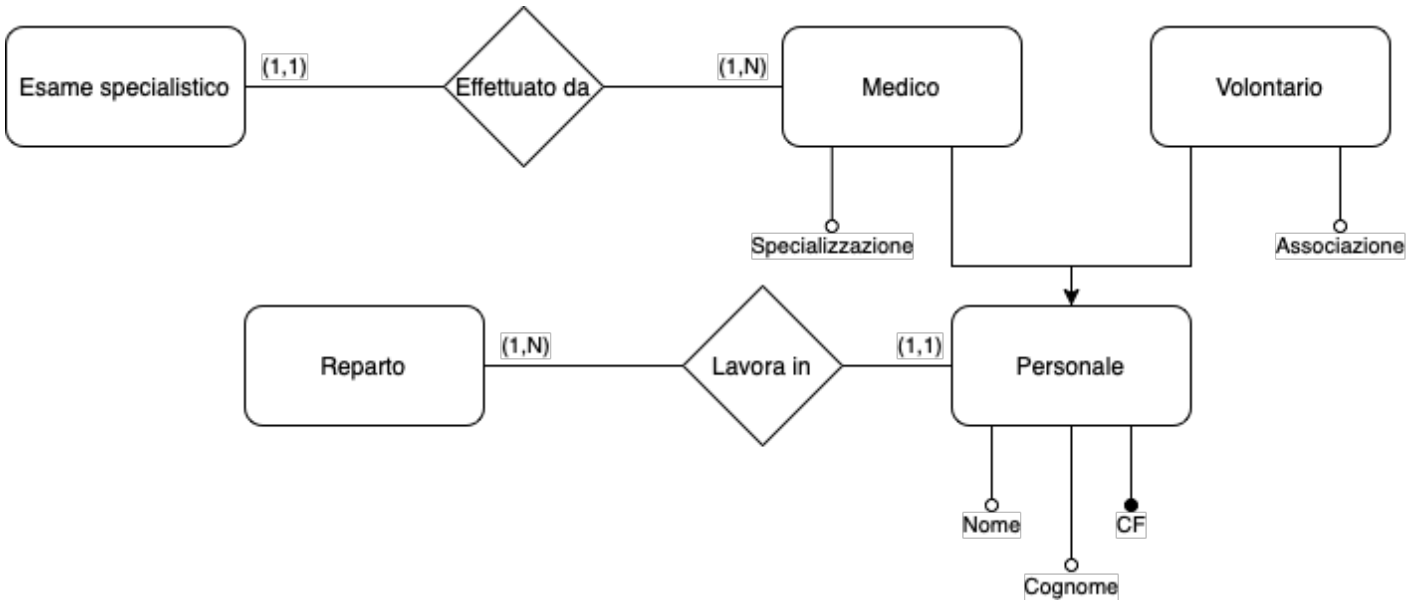
L'entità padre può generalizzare diverse sottoentità rispetto ad un unico criterio. In questo caso si parla di **generalizzazione**.

Nella generalizzazione, le sottoentità hanno insiemi di istanze disgiunti a coppie (anche se in alcune varianti del modello ER, si può specificare se due sottoentità della
stessa entità padre sono disgiunte o no).



Modello ER - Generalizzazione

L'entità padre può generalizzare diverse sottoentità rispetto ad un unico criterio. In questo caso si parla di **generalizzazione**.



Modello ER - Generalizzazione

Il principio di ereditarietà vale anche per le generalizzazioni:

- ogni proprietà dell'entità è padre è anche una proprietà della sottoentità, e non si riporta esplicitamente nel diagramma
- l'entità figlia può avere ovviamente ulteriori proprietà.

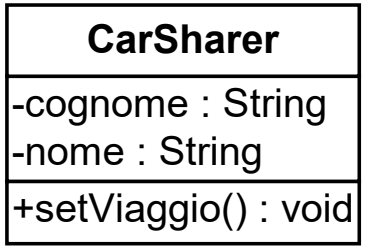
Vige la regola che una entità può avere al massimo una entità padre. In altre parole, il modello ER non ammette ereditarietà multipla

UML-Diagramma delle Classi

Definizione: il *diagramma delle classi* è un grafo che descrive i tipi degli oggetti in un sistema, le relazioni statiche tra essi, gli attributi e le operazioni di una classe, ed i vincoli sulle relazioni

Una classe è rappresentata da un rettangolo scomposto in tre parti:

- il nome della classe
- gli attributi della classe
- le operazioni della classe

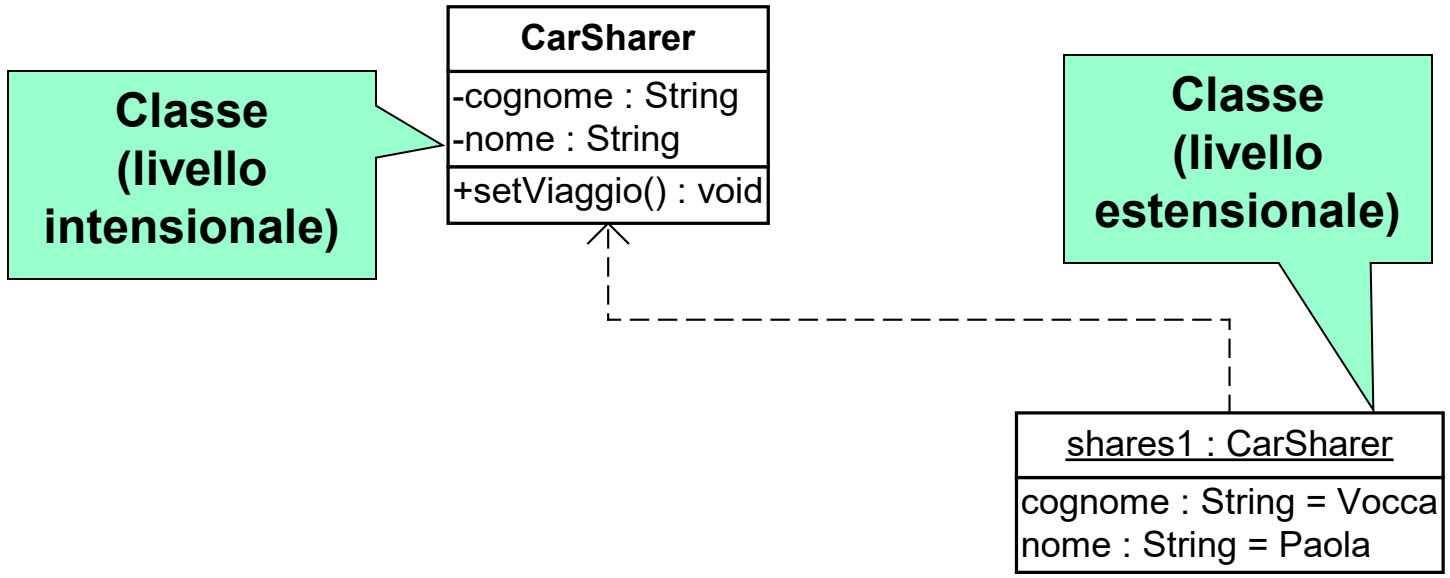


Proprietà di una classe: Attributi in UML

- Un attributo modella una proprietà locale della classe ed è caratterizzato da un nome e dal tipo dei valori associati
- Ogni attributo di una classe stabilisce una proprietà locale valida per tutte le istanze della classe. Il fatto che la proprietà sia locale significa che è un proprietà indipendente da altri oggetti
- Formalmente, un attributo *A* della classe *C* si può considerare una funzione che associa un valore di tipo *T* ad ogni oggetto che è istanza di *C*

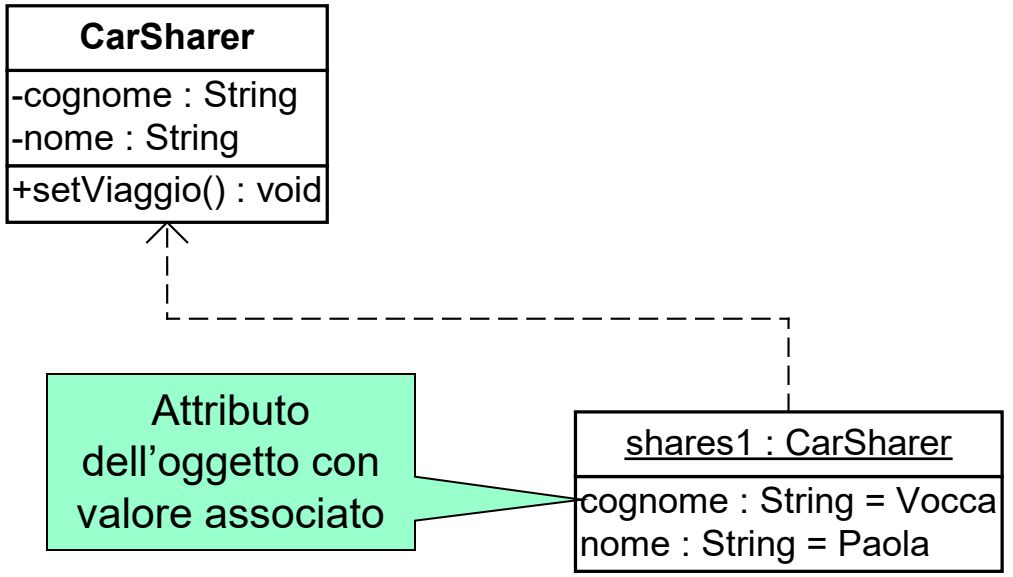
Istanze di una classe

- Tra un oggetto che è istanza di una classe *C* e la classe *C* si traccia un arco Instance-of (l'arco in realtà non è strettamente necessario, perchè la classe di cui l'oggetto è istanza è già indicata nell'oggetto)
- Ricordiamo che gli oggetti formano il livello estensionale, mentre le classi a livello intensionale.



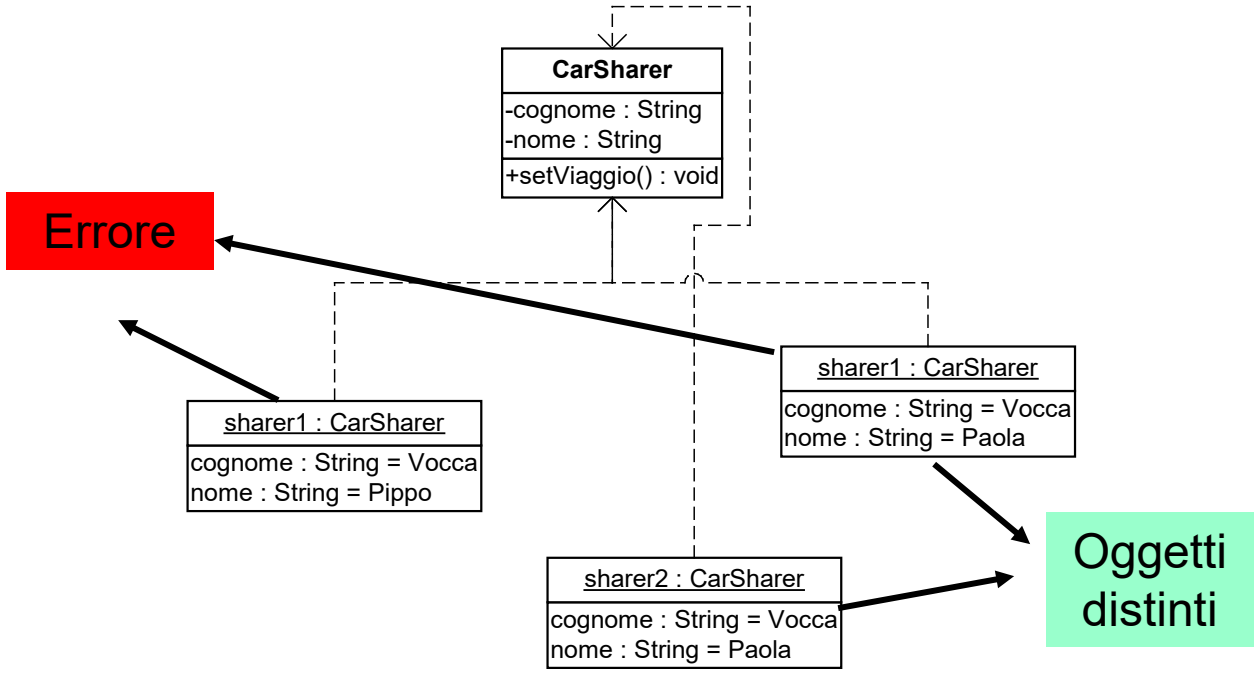
Attributi di oggetti

- Gli attributi di una classe determinano gli attributi delle sue istanze
- **Regola importante**: se una classe *C* ha un attributo *A* di tipo *T*, **ogni** oggetto che è istanza di *C* ha l'attributo *A*, con un valore associato di tipo *T*
- **Regola importante**: un oggetto X non può avere un valore per un attributo che non è definito nella classe di cui X è istanza



Identificatori degli oggetti

- Due oggetti con identificatori distinti sono comunque distinti, anche se hanno i valori di tutti gli attributi uguali
- Due oggetti diversi devono avere identificatori diversi, anche se possono avere gli stessi valori per tutti gli attibuti



I diagrammi delle classi (1)

- Secondo la metodologia UML vengono definiti come *Diagrammi a struttura statica*. Vengono utilizzati per:
  - Documentare le classi che compongono un sistema o un sottosistema.
  - Descrivere ***associazioni***, ***generalizzazioni aggregazioni***, fra le varie classi.
  - Evidenziare le caratteristiche di una classe - ***attributi*** e ***operazioni***

I diagrammi delle classi (2)

- I diagrammi delle classi possono essere utilizzati in varie fasi dello sviluppo di un sistema:
  - In fase di ***analisi***: per la specifica delle classi all'interno del dominio del problema.
  - In fase di ***progettazione***: per la rappresentazione delle classi e relazioni che riflettono il modello della soluzione.
- Possono documentare come interagiscono le classi di un particolare sistema con le librerie di classi già esistenti.
- Possono essere utilizzati a rappresentare istanze di oggetti all'interno delle classi.
- Possono mostrare le interfacce di una classe.

Tipi di attributi e di operazioni (1)

- *Sintassi*

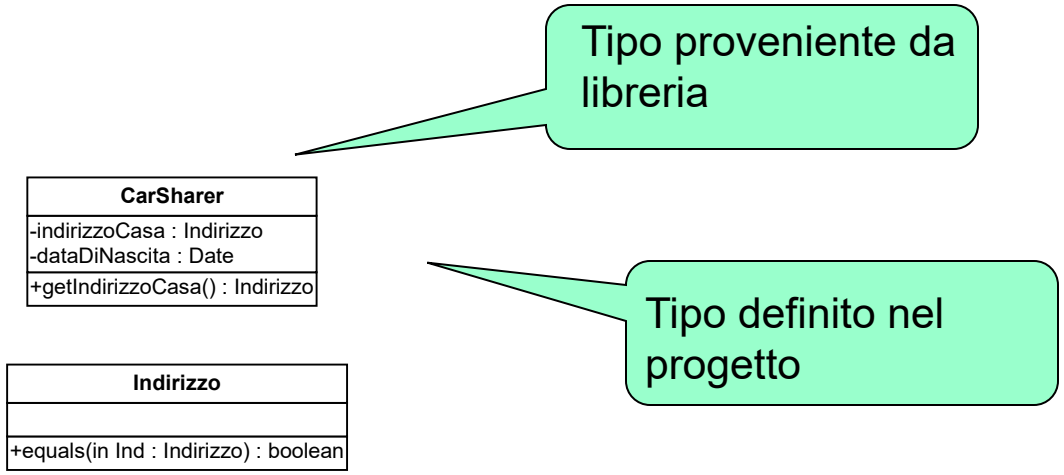
  <nomeCaratteristica>:<tipo>

- *Semantica*
  - ***nomeCaratteristica*** identifica o un attributo oppure un'operazione
  - ***Tipo*** identifica il tipo di dato dell'attributo oppure il tipo di dato restituito dall'operazione

    <nomeCaratteristica>:<tipo>

- ***N.B.*** Gli attributi e le operazioni possono essere tipizzati come classi provenienti:
  - Dalle librerie dell'ambiente d'implementazione.
  - Dal modello delle classi in uso.

Tipi di attributi e di operazioni (2)

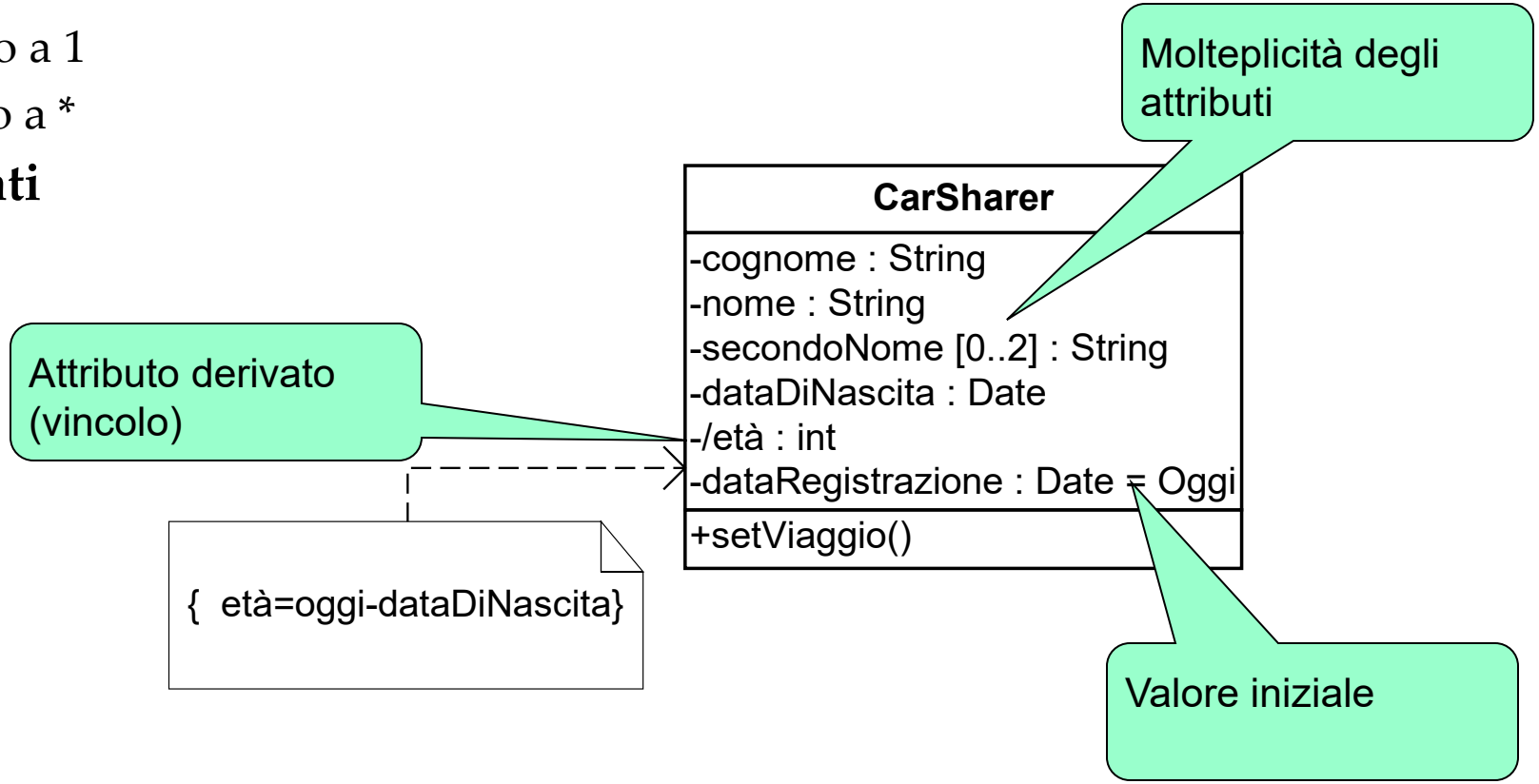


Attributi e operazioni: Visibilità

- **- private**:disponibile solo all'interno della classe che la definisce
- **+ public**: disponibile solo per le classi associate alla classe che la definisce;
- # **protected**: disponibile solo all'interno della classe che la possiede e di ogni sua sottoclasse.

Attributi: ulteriori specifiche

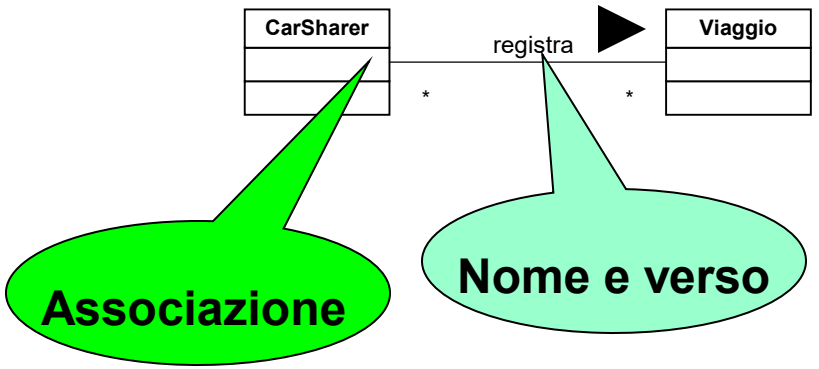
- **Molteplicità**
  - [m..n] dove: m numero minimo e n numero massimo
    - 1..1 troncato a 1
    - 0..* troncato a *
- **Attributi derivati**
- **Valori iniziali**



Associazioni

- Per il momento, ci limitiamo a discutere associazioni tra due classi (ma le associazioni possono coinvolgere N classi)
- Una ***associazione*** (o relazione) tra una classe *C1* ed una classe *C2* modella una relazione matematica tra l'insieme delle istanze di *C1* e l'insieme delle istanze di *C2*
- Gli attributi modellano proprietà locali di una classe, le associazioni modellano proprietà che coinvolgono altre classi.
- Una associazione tra due classi modella una proprietà di entrambe le classi

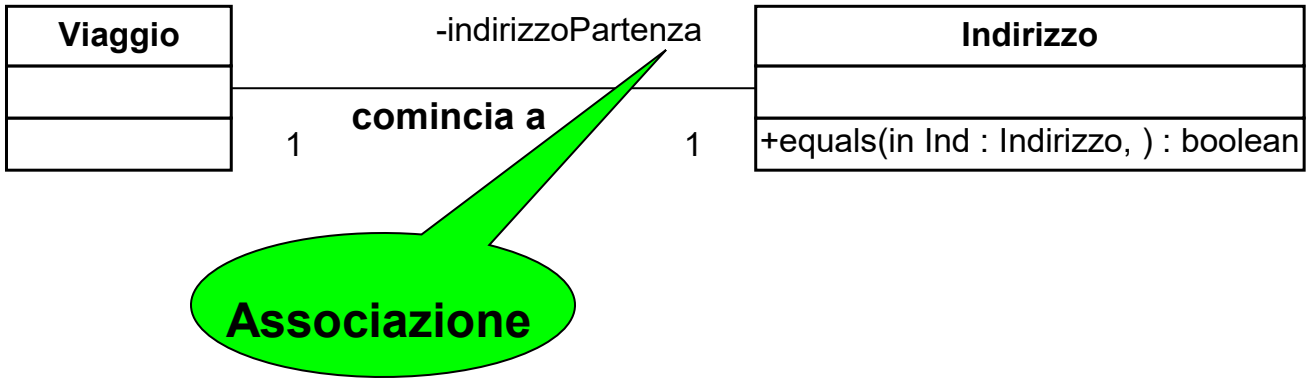
Associazione: Esempio



- Alcune volte è interessante specificare un verso per il nome della associazione
- **Attenzione**: la notazione riportata sopra **non** significa che l'associazione ha un verso. In altre parole, il verso non è una caratteristica del significato della associazione, ma dice semplicemente che il nome scelto per la associazione evoca un verso

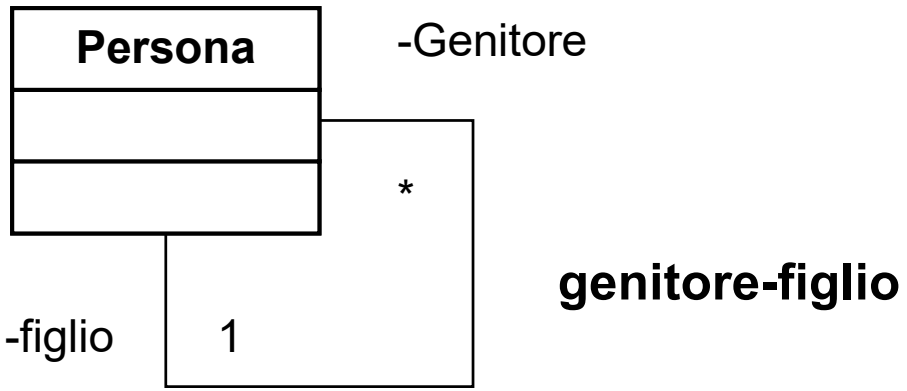
Associazione: Ruoli (1)

- È possibile aggiungere alla associazione una informazione che specifica il *ruolo* che una classe gioca nella associazione
- Il ruolo si indica con un nome posizionato lungo la linea che rappresenta l'associazione, vicino alla classe alla quale si riferisce



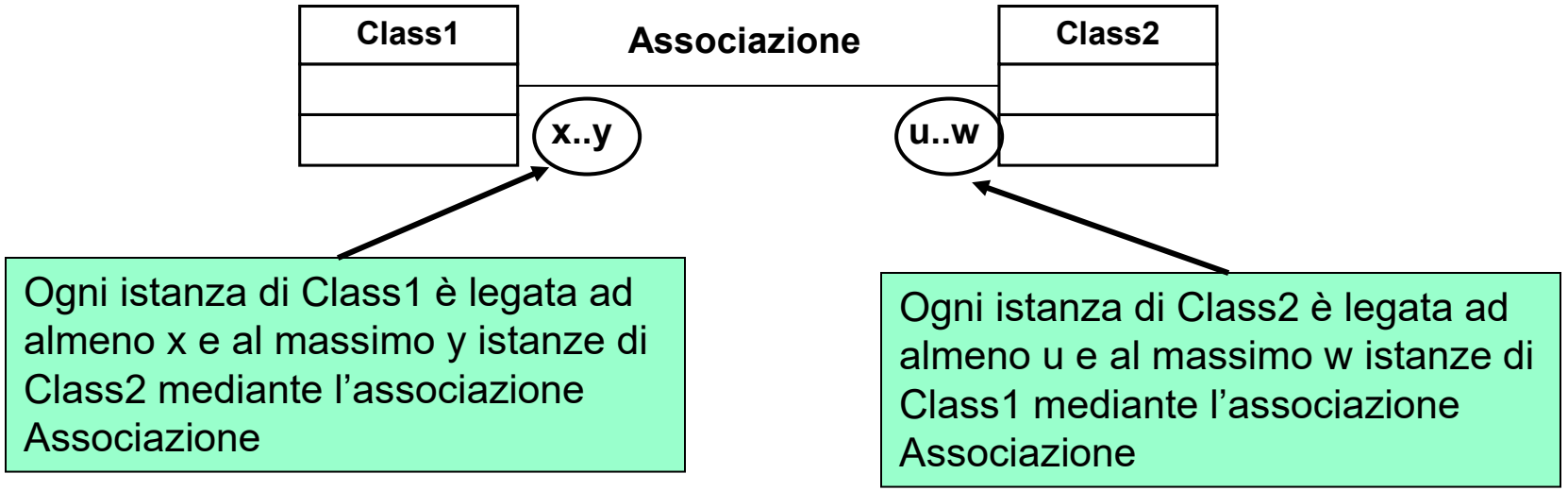
Associazione: Ruoli (2)

- Analogamente al verso, il ruolo è generalmente opzionale, e non aggiunge nulla al significato del'associazione
- L'unico caso in cui **il ruolo è obbligatorio è quello in cui l'associazione insiste più volte sulla stessa classe**, e rappresenta una relazione non simmetrica

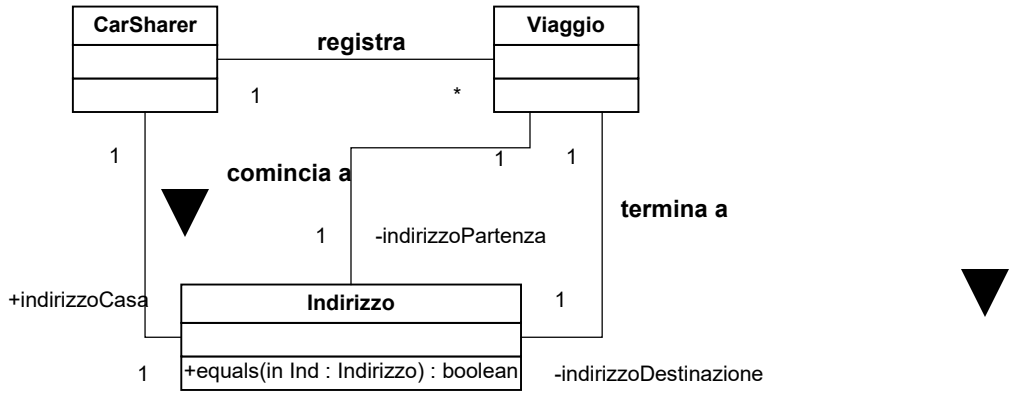


Associazione: Molteplicità

- Per specificare con maggiore precisione il significato delle associazioni binarie si possono definire i vincoli di ***molteplicità*** (o semplicemente molteplicità) delle associazioni

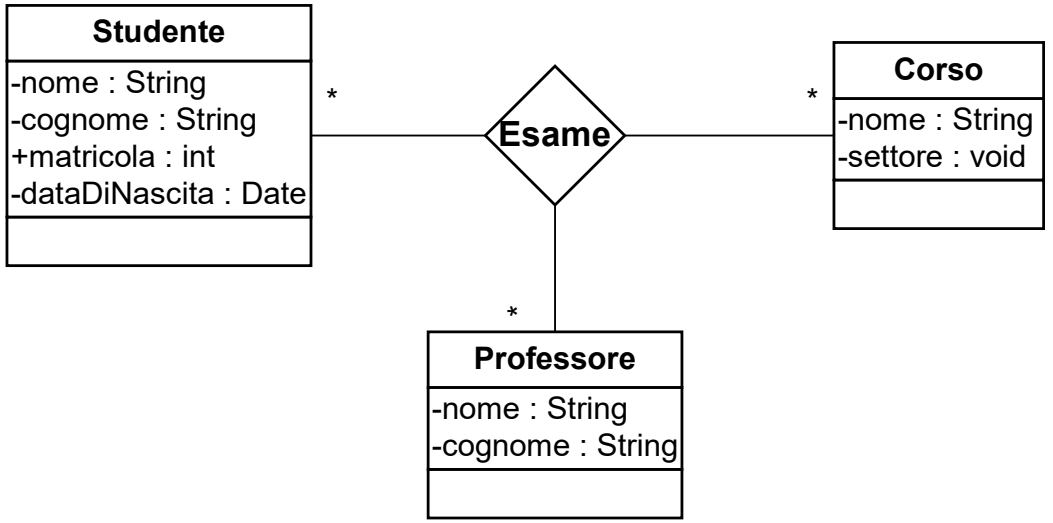


Associazione: Molteplicità (2)



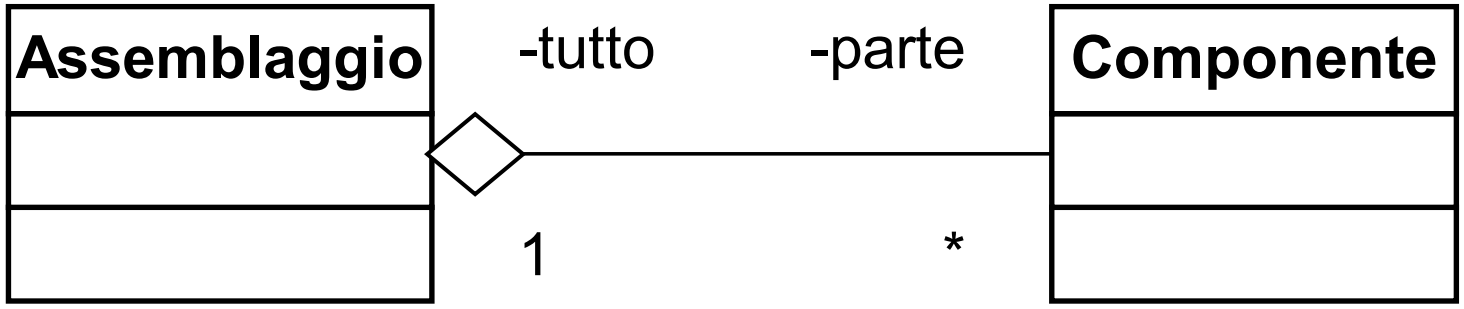
Associazioni N-arie

- Una associazione può essere definita su tre o più classi. In tale caso ***l'associazione si dice n-aria***, modella una relazione matematica tra n insiemi



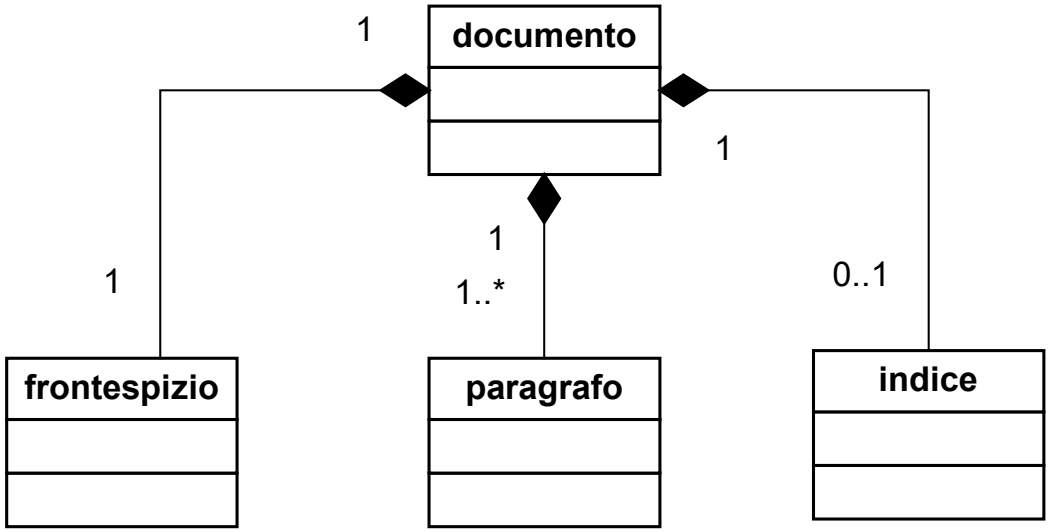
Aggregazioni in UML

- Un'*aggregazione* viene utilizzata per indicare che, oltre ad avere attributi propri, l'istanza di una classe può consistere di, o comprendere, istanze di altre classi
- Per riferirsi alle aggregazione si utilizza il termine ***tutto-parte.***



Composizioni in UML

- A differenza delle aggregazioni, in cui le istanza componenti possono esistere a prescindere dalla istanza di cui sono componenti, in una ***composizione*** le parti non possono esistere senza la parte "tutto".

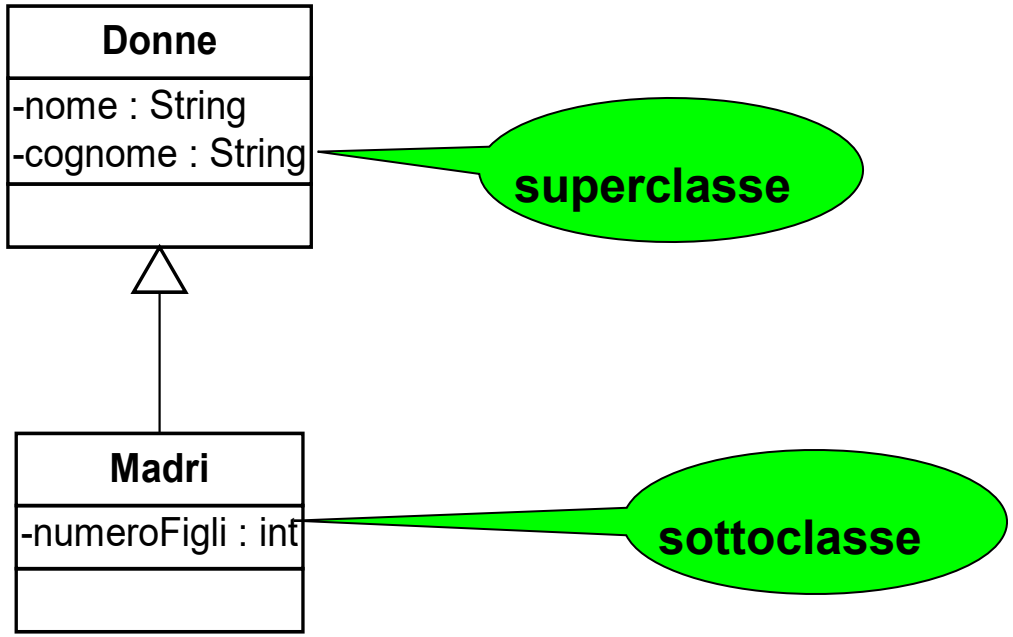


Generalizzazioni in UML (1)

- Fino ad ora abbiamo assunto che due classi siano sempre disgiunte. In realtà sappiamo che può accadere che tra due classi sussista la relazione is-a, e cioè che ogni istanza di una sia anche istanza dell'altra
- In UML la relazione is-a si modella mediante la nozione di ***generalizzazione***
- La generalizzazione coinvolge una superclasse ed una o più sottoclassi (dette anche ***classi derivate***). Il significato della generalizzazione è il seguente: ogni istanza di ciascuna sottoclasse è anche istanza della superclasse
- Quando la sottoclasse è una, la generalizzazione modella appunto la relazione is-a tra la sottoclasse e la superclasse

Generalizzazioni in UML (2)

- Esempio di generalizzazione (siccome la generalizzazione coinvolge due classi, essa modella la relazione is-a):

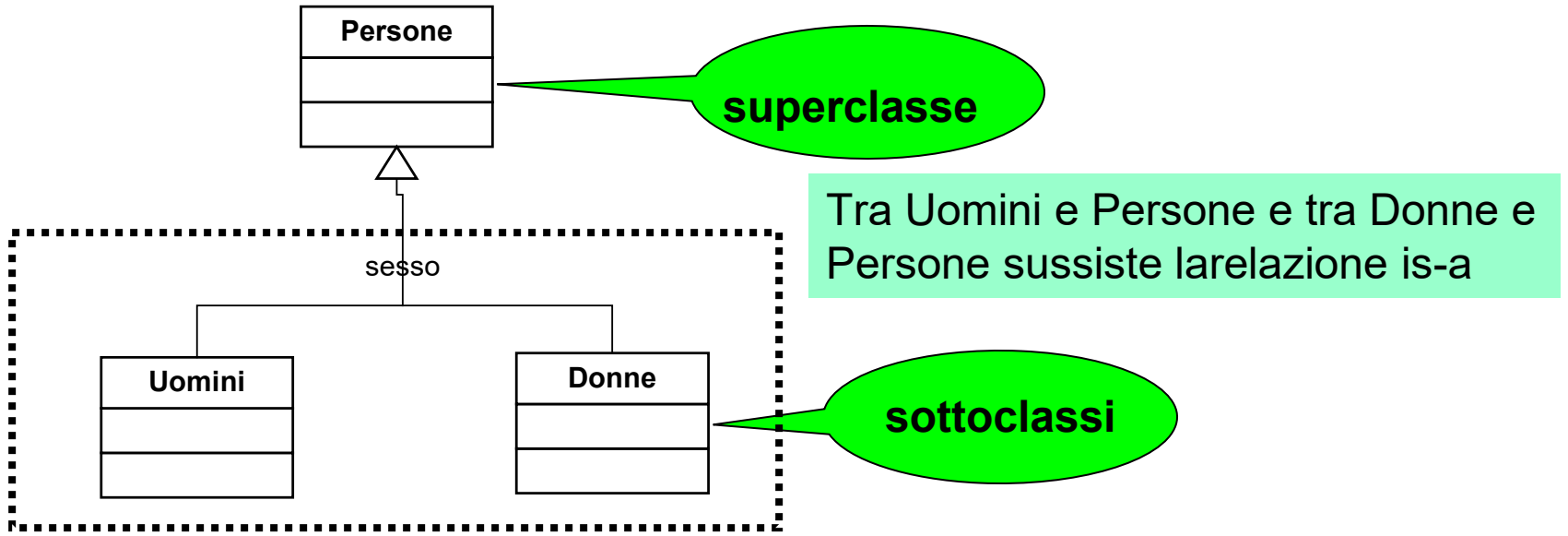


Generalizzazioni in UML (3)

- ***Principio di ereditarietà***: ogni proprietà della superclasse è anche una proprietà della sottoclasse, e non si riporta esplicitamente nel diagramma

Generalizzazioni in UML (4)

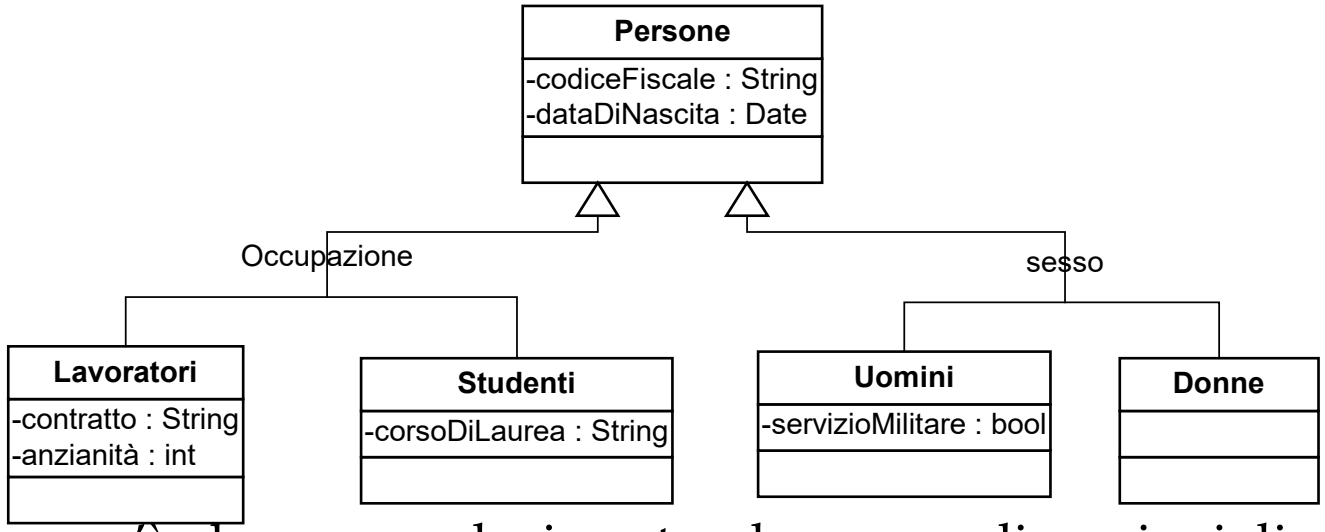
- Le ***generalizzazione*** oltre a modellare la relazione "is-a" fra due classi, vengono utilizzate anche per rappresentare la relazione gerarchica fra più classi rispetto ad un unico criterio (che si può indicare con un nome, che è il nome della generalizzazione.



Tra Uomini e Persone e tra Donne e Persone sussiste larelazione is-a

Diverse generalizzazioni di una stessa classe

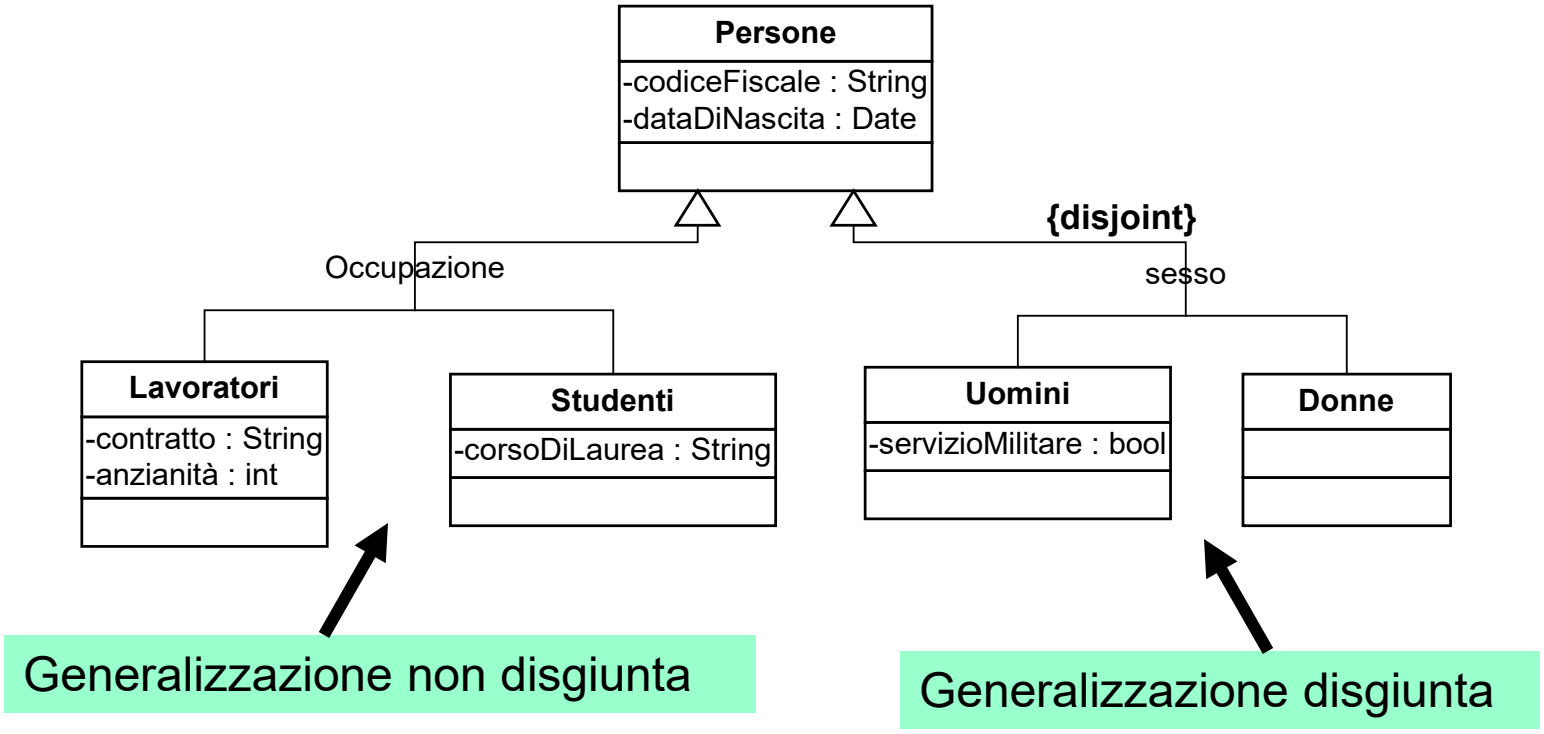
- La stessa superclasse può partecipare a diverse generalizzazioni



- Concettualmente, non c'è alcuna correlazione tra due generalizzazioni diverse, perché rispondono a due criteri diversi di classificare le istanze della superclasse

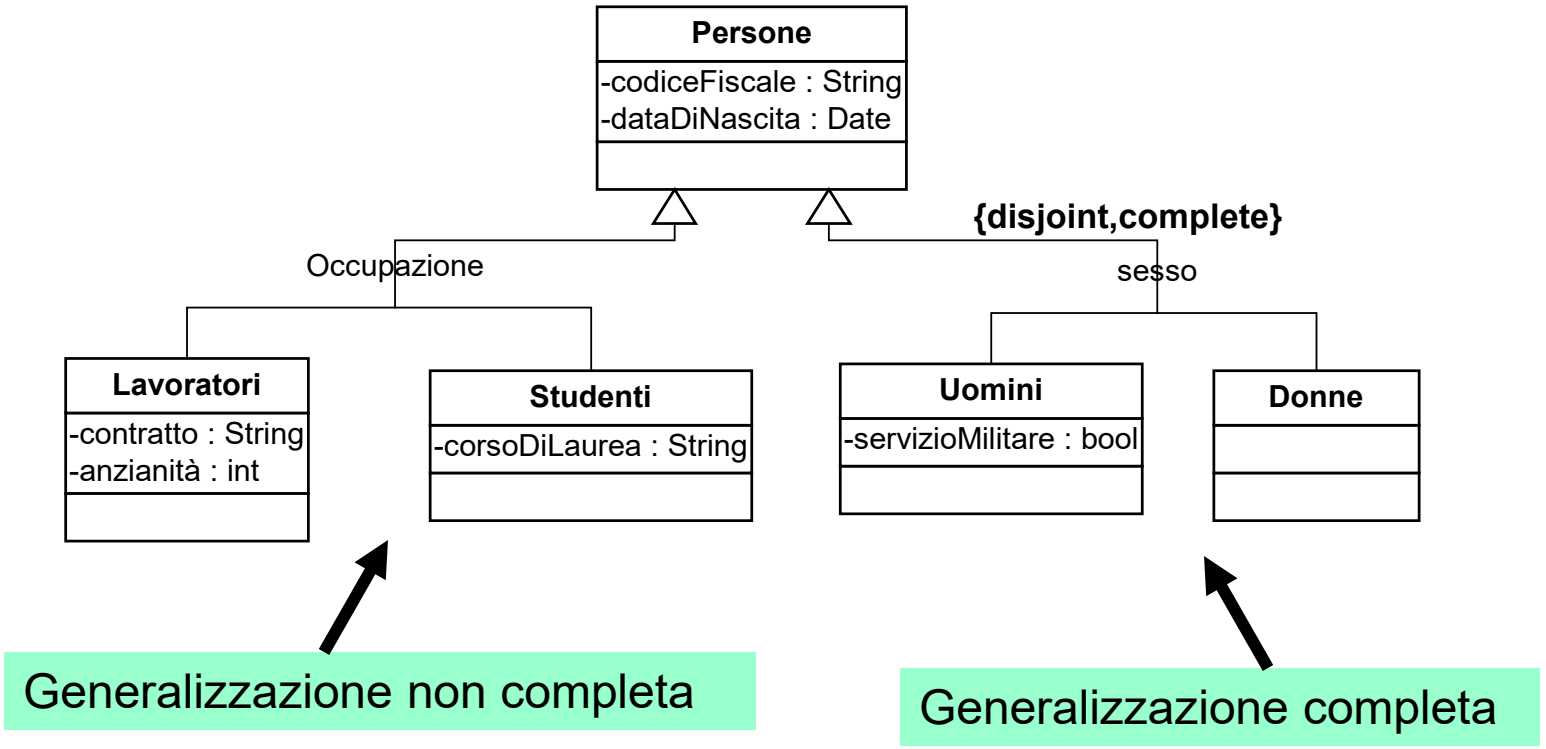
Generalizzazioni `disjoint`

- Una generalizzazione può essere disgiunta (le sottoclassi sono disgiunte a coppie) o no



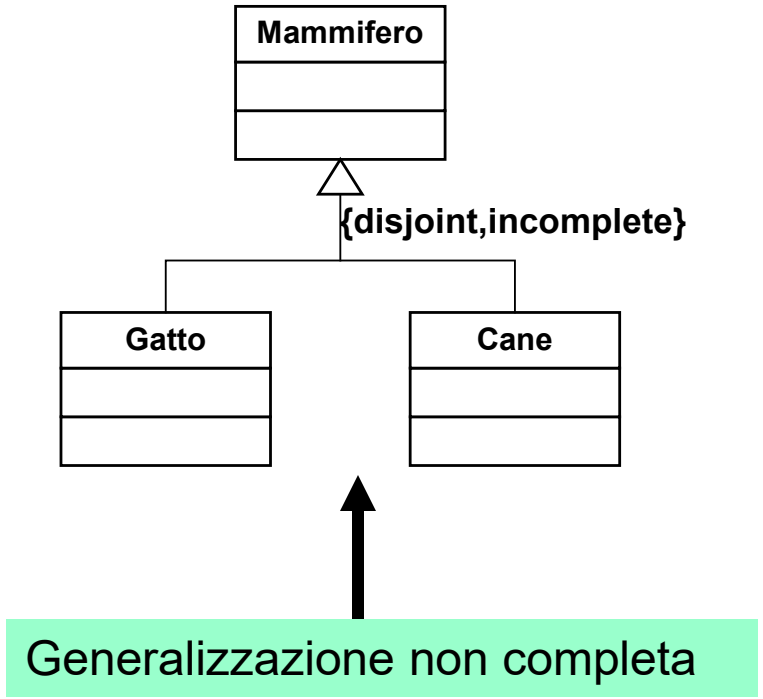
Generalizzazioni complete

- Una generalizzazione può essere completa (l'unione delle istanze delle sottoclassi è uguale all'insieme delle istanze della superclasse) o no.



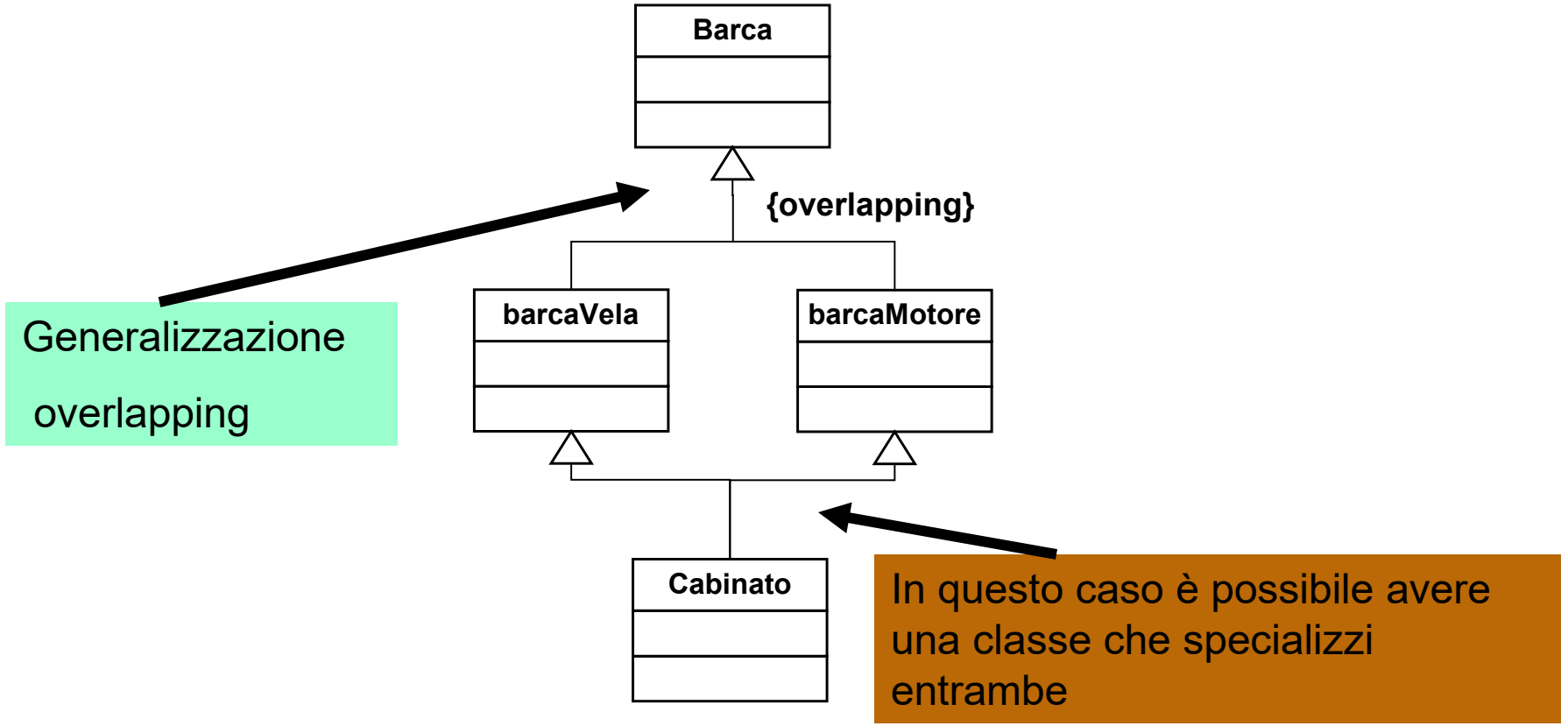
Generalizzazioni `incomplete`

- Una generalizzazione ***incomplete*** è tale che l'unione delle istanze delle sottoclassi non è uguale all'insieme delle istanze della superclasse .



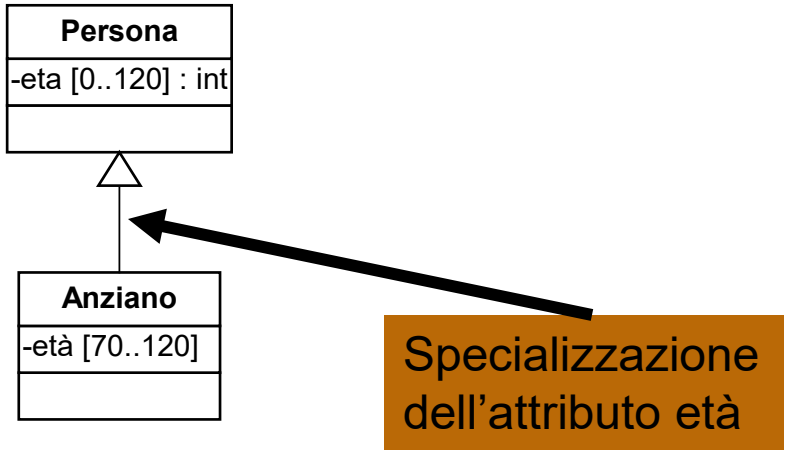
Generalizzazioni `overlapping`

- Una generalizzazione ***overlapping*** è l'opposto di quella disjoint. Un'istanza può esserlo di più di una delle sottoclassi della superclasse



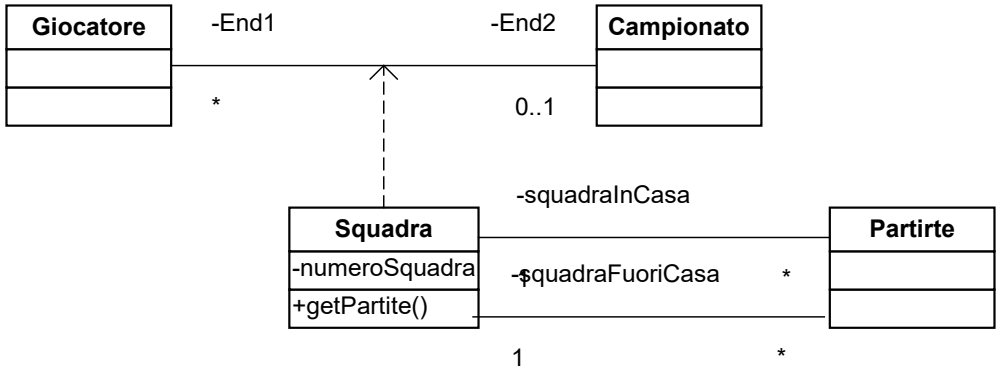
Specializzazioni

- In una generalizzazione la sottoclasse non solo può avere proprietà aggiuntive rispetto alla superclasse, ma può anche specializzare le proprietà ereditate dalla superclasse.
  - Specializzazione di un attributo: Se una classe *C1* ha un attributo *A* di tipo *T1*, e se *C2* è una sottoclasse di *C1*, specializzare *A* in *C2* significa definire *A* anche in *C2* ed assegnargli un tipo *T2* i cui valori sono un sottoinsieme dei valori di *T*.



Classi di associazoni

- Le **Classi associazioni** permettono di esplicitare gli attributi e le associazioni di una associazione.



UML Example for Displaying Specialization / Generalization