

# **Basi di dati**

**Progettazione di basi di dati:  
Metodologie e modelli**

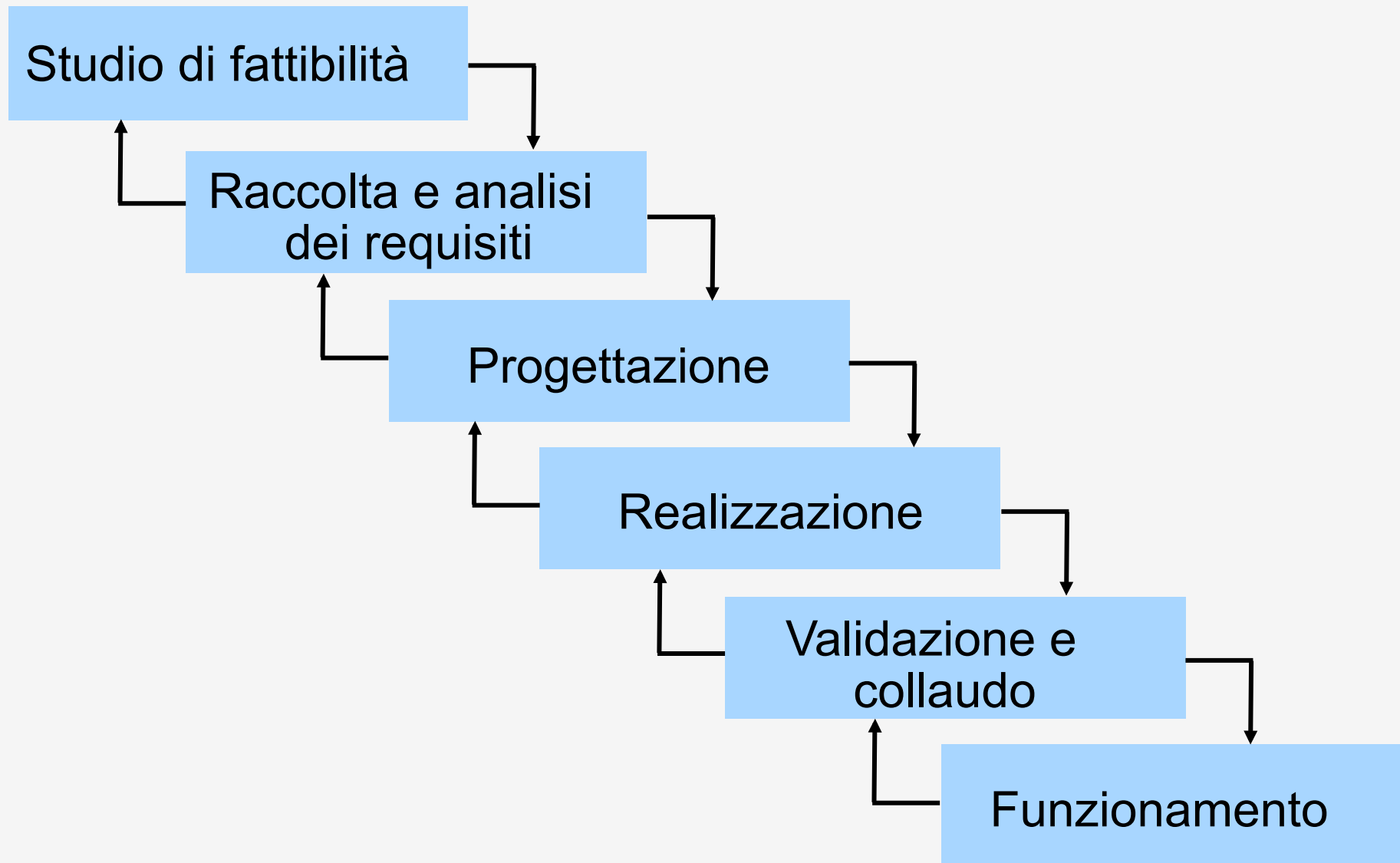
## Perché preoccuparci?

- Proviamo a modellare una applicazione definendo direttamente lo schema logico della base di dati:
  - da dove cominciamo?
  - rischiamo di perderci subito nei dettagli
  - dobbiamo pensare subito a come correlare le varie tabelle (chiavi etc.)
  - il modello relazionale è “rigido”

# Progettazione di basi di dati

- È una delle attività del processo di sviluppo dei sistemi informativi
- va quindi inquadrata in un contesto più generale:
- **il ciclo di vita dei sistemi informativi:**
  - Insieme e sequenzializzazione delle attività svolte da analisti, progettisti, utenti, nello sviluppo e nell'uso dei sistemi informativi
  - attività iterativa, quindi **ciclo**

## Il ciclo di vita



## **Fasi (tecniche) del ciclo di vita**

- Studio di fattibilità: definizione costi e priorità
- Raccolta e analisi dei requisiti: studio delle proprietà del sistema
- Progettazione: di dati e funzioni
- Realizzazione
- Validazione e collaudo: sperimentazione
- Funzionamento: il sistema diventa operativo

# Progettazione

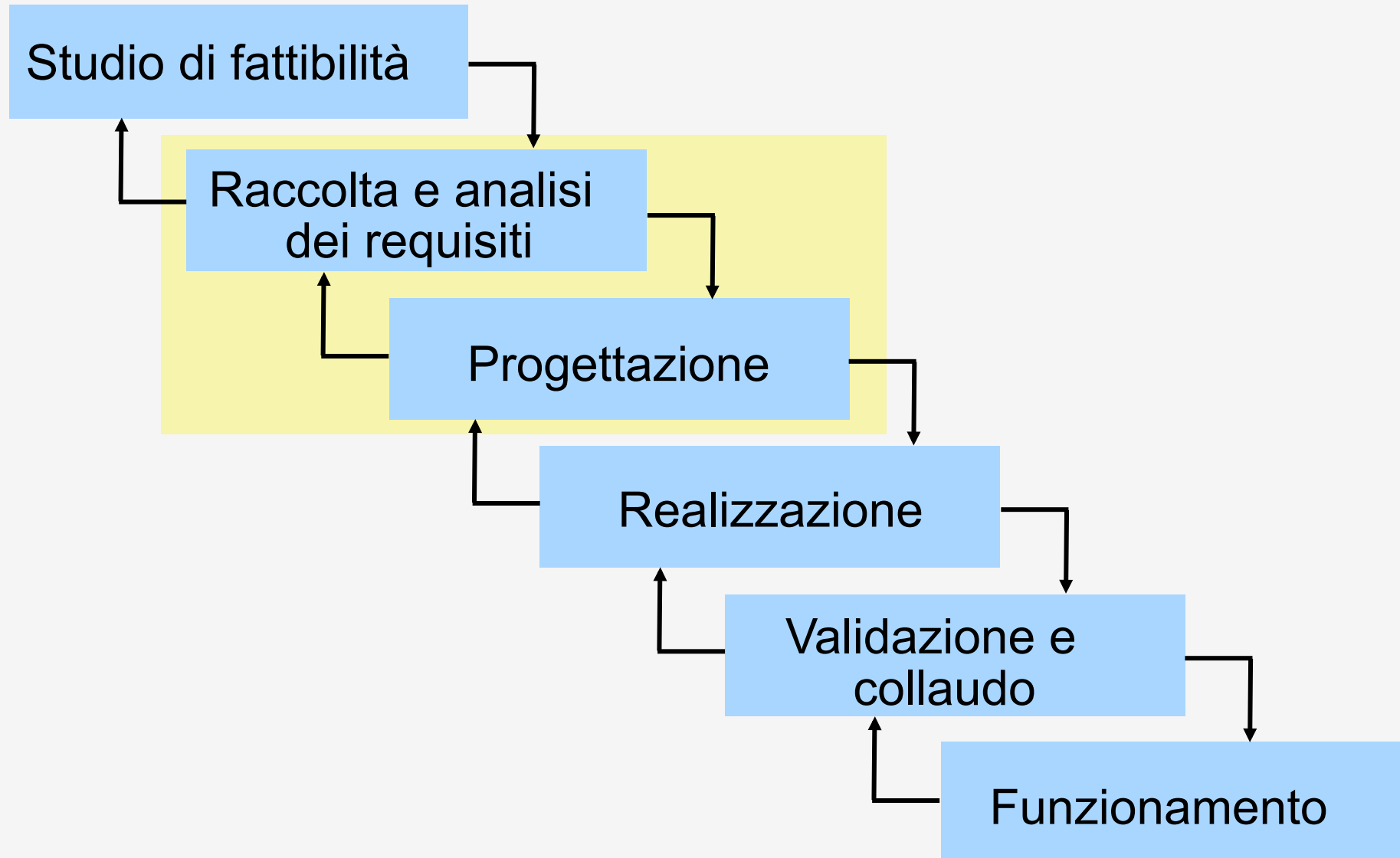
La progettazione di un sistema informativo riguarda due aspetti:

- ▶ **progettazione dei dati**
- ▶ **progettazione delle applicazioni**

**Ma:**

- ▶ **i dati hanno un ruolo centrale**
  - **i dati sono più stabili**

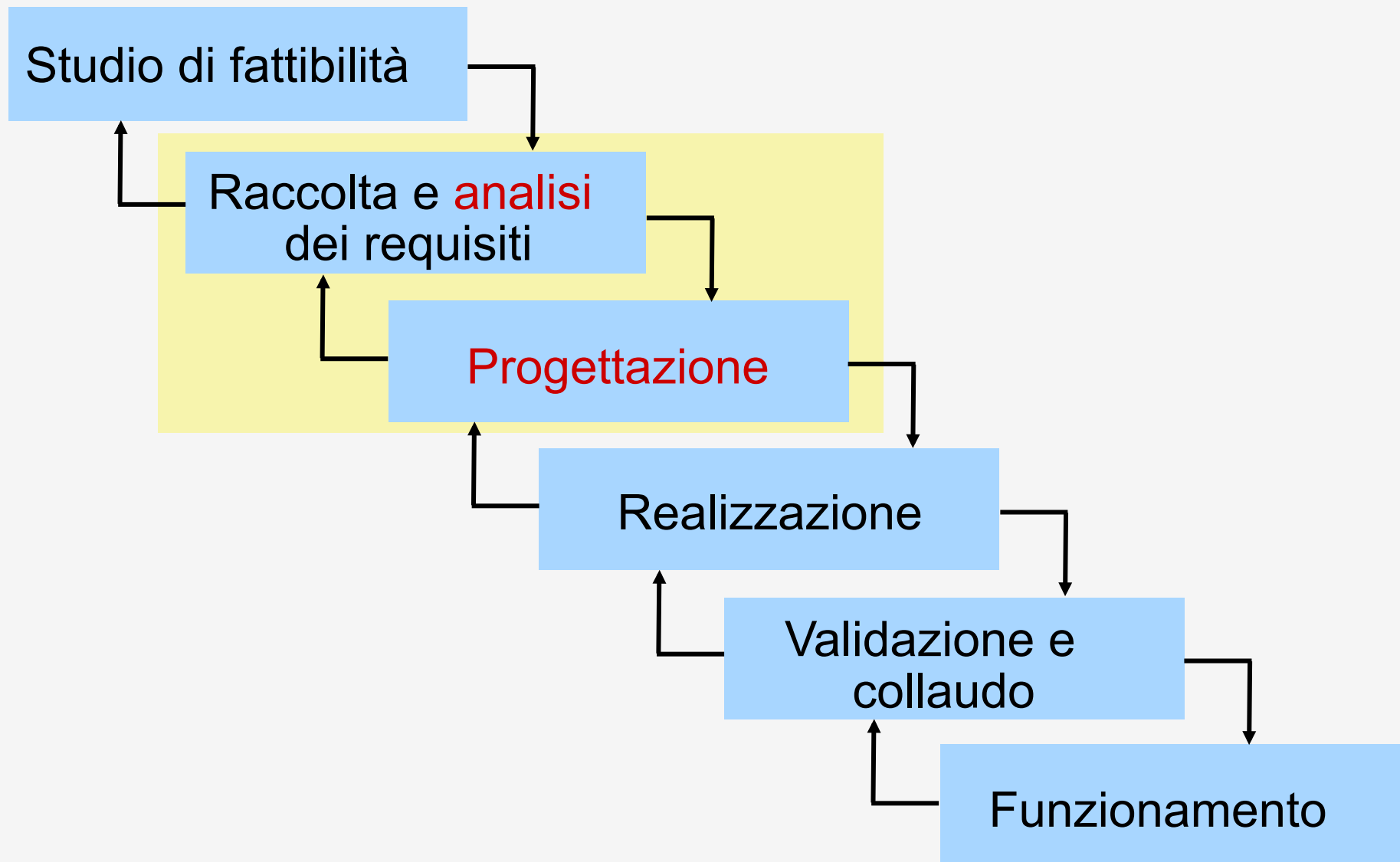
# Analisi & Progettazione



# Metodologia di progetto

- Per garantire prodotti di buona qualità è opportuno seguire una
  - **metodologia di progetto**, con:
    - articolazione delle attività in fasi
    - criteri di scelta
    - modelli di rappresentazione
    - generalità e facilità d'uso





# Analisi & Progettazione: dettagli

Requisiti della base di dati

Progettazione  
concettuale

**“CHE COSA”:**  
**analisi**

Schema concettuale

Progettazione  
logica

Schema logico

**“COME”:**  
**progettazione**

Progettazione  
fisica

Schema fisico

# Modelli di dati: concettuale, logico e fisico

I prodotti della varie fasi sono schemi di alcuni  
modelli di dati:

- Schema concettuale
- Schema logico
- Schema fisico

# 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 nuovi tipi, 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

# Schemi e istanze

- In ogni base di dati esistono:
  - lo **schema**, sostanzialmente invariante nel tempo, che ne descrive la struttura (aspetto intensionale)
    - nel modello relazionale, le intestazioni delle tabelle
  - l'**istanza**, i valori attuali, che possono cambiare anche molto rapidamente (aspetto estensionale)
    - nel modello relazionale, il “corpo” di ciascuna tabella

## Due tipi (principali) di modelli

- **modelli logici**: utilizzati nei DBMS esistenti per l'organizzazione dei dati
  - utilizzati dai programmi
  - indipendenti dalle strutture fisicheesempi: **relazionale**, reticolare, gerarchico, a oggetti
- **modelli concettuali**: permettono di rappresentare i dati in modo indipendente da ogni sistema
  - cercano di descrivere i concetti del mondo reale
  - sono utilizzati nelle fasi preliminari di progettazioneil più noto è il modello **Entity-Relationship**

## Modelli concettuali, perché?

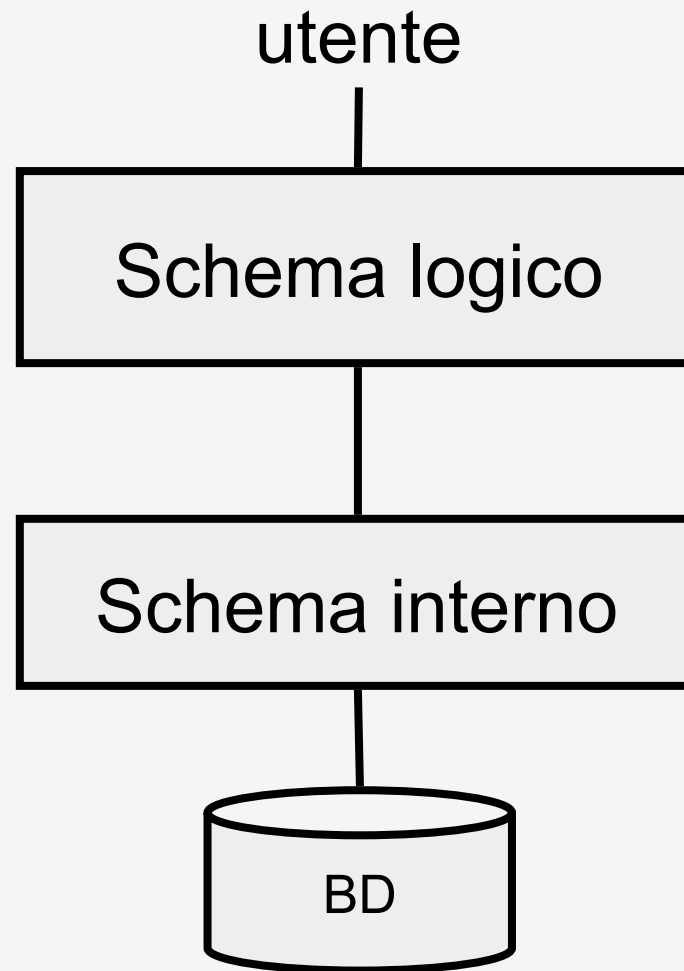
- Proviamo a modellare una applicazione definendo direttamente lo schema logico della base di dati:
  - da dove cominciamo?
  - rischiamo di perderci subito nei dettagli
  - dobbiamo pensare subito a come correlare le varie tabelle (chiavi etc.)
  - i modelli logici sono rigidi

## Modelli concettuali, perché?

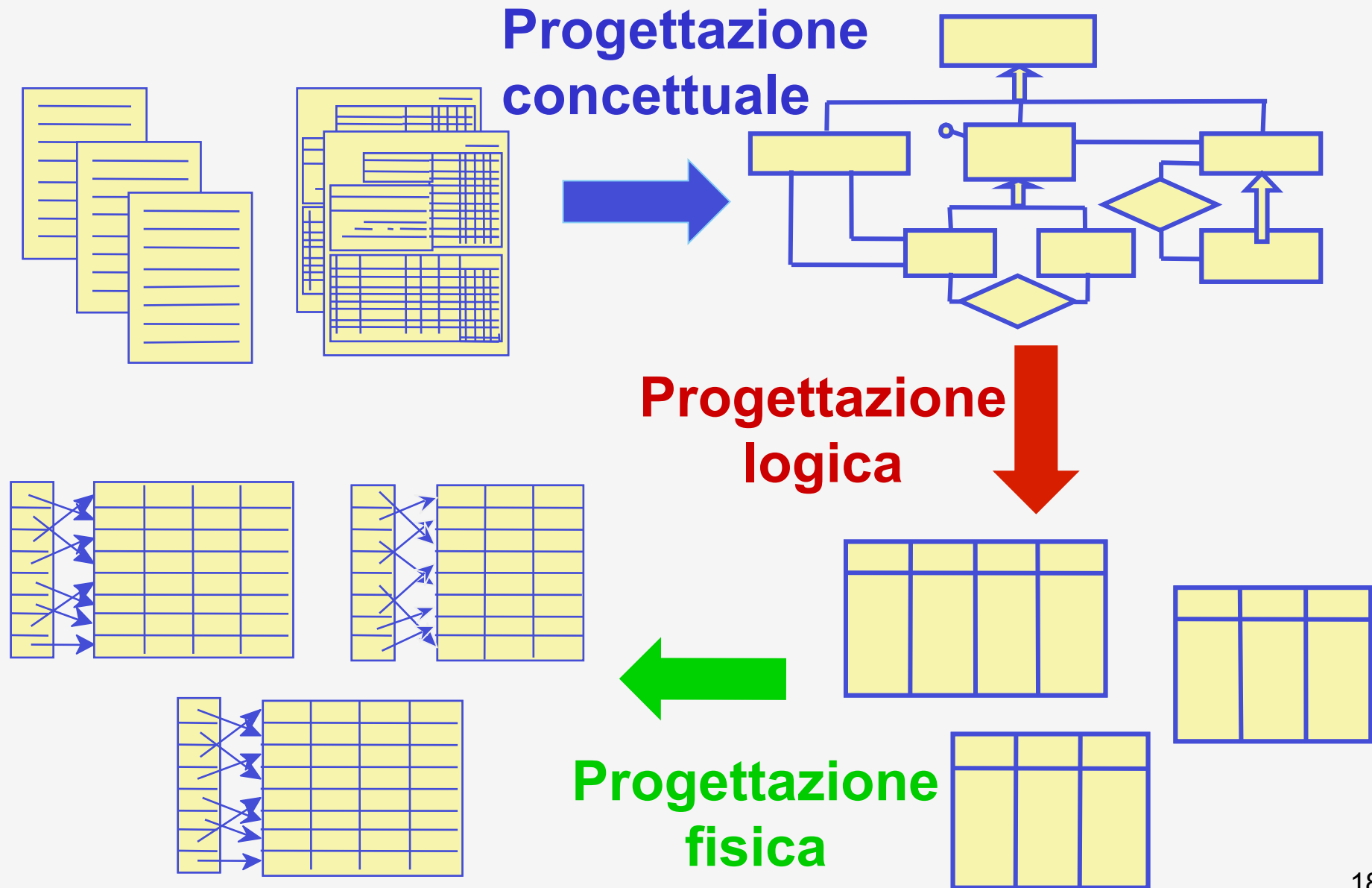
- servono per ragionare sulla realtà di interesse, indipendentemente dagli aspetti realizzativi
- permettono di rappresentare le classi di oggetti di interesse e le loro correlazioni
- prevedono efficaci rappresentazioni grafiche (utili anche per documentazione e comunicazione)



# Architettura (semplificata) di un DBMS



# Progettazione: evoluzione



# **Modello Entity-Relationship (Entità-Relazione)**

- Il più diffuso modello concettuale
  - Ne esistono molte versioni,
  - (più o meno) diverse l'una dall'altra

# I costrutti del modello E-R

- Entità
- Relationship
- Attributo
- Identificatore
- Generalizzazione
- ....

# Entità

- Classe di oggetti (fatti, persone, cose) della realtà di interesse con proprietà comuni e con esistenza “autonoma”
- Esempi:
  - impiegato, città, conto corrente, ordine, fattura

# Relationship

- Legame logico fra due o più entità, rilevante nell'applicazione di interesse
- Esempi:
  - Residenza (fra persona e città)
  - Esame (fra studente e corso)

# Uno schema E-R, graficamente



## Entità (2)

- Classe di oggetti (fatti, persone, cose) della realtà di interesse con proprietà comuni e con esistenza “autonoma”
- Esempi:
  - impiegato, città, conto corrente, ordine, fattura



# Entità: schema e istanza

- Entità:
  - classe di oggetti, persone, ... "omogenei"
- Occorrenza (o istanza) di entità:
  - elemento della classe (l'oggetto, la persona, ..., non i dati)
- nello schema concettuale rappresentiamo le entità, non le singole istanze ("astrazione")

# Rappresentazione grafica di entità

**Impiegato**

**Dipartimento**

**Città**

**Vendita**

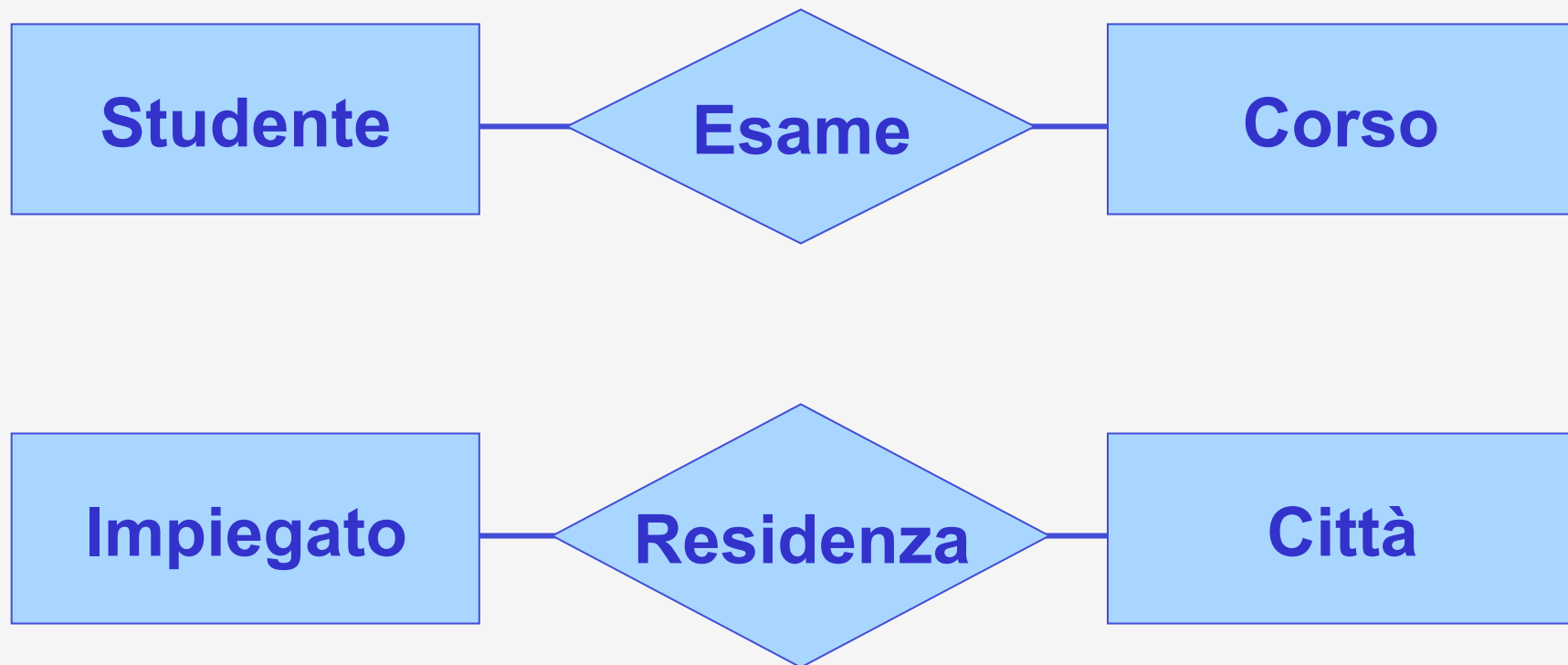
# Entità, commenti

- Ogni entità ha un nome che la identifica univocamente nello schema:
  - nomi espressivi
  - opportune convenzioni
    - singolare

## Relationship (2)

- Legame logico fra due o più entità, rilevante nell'applicazione di interesse
- Esempi:
  - Residenza (fra persona e città)
  - Esame (fra studente e corso)
- Chiamata anche:
  - relazione, correlazione, associazione

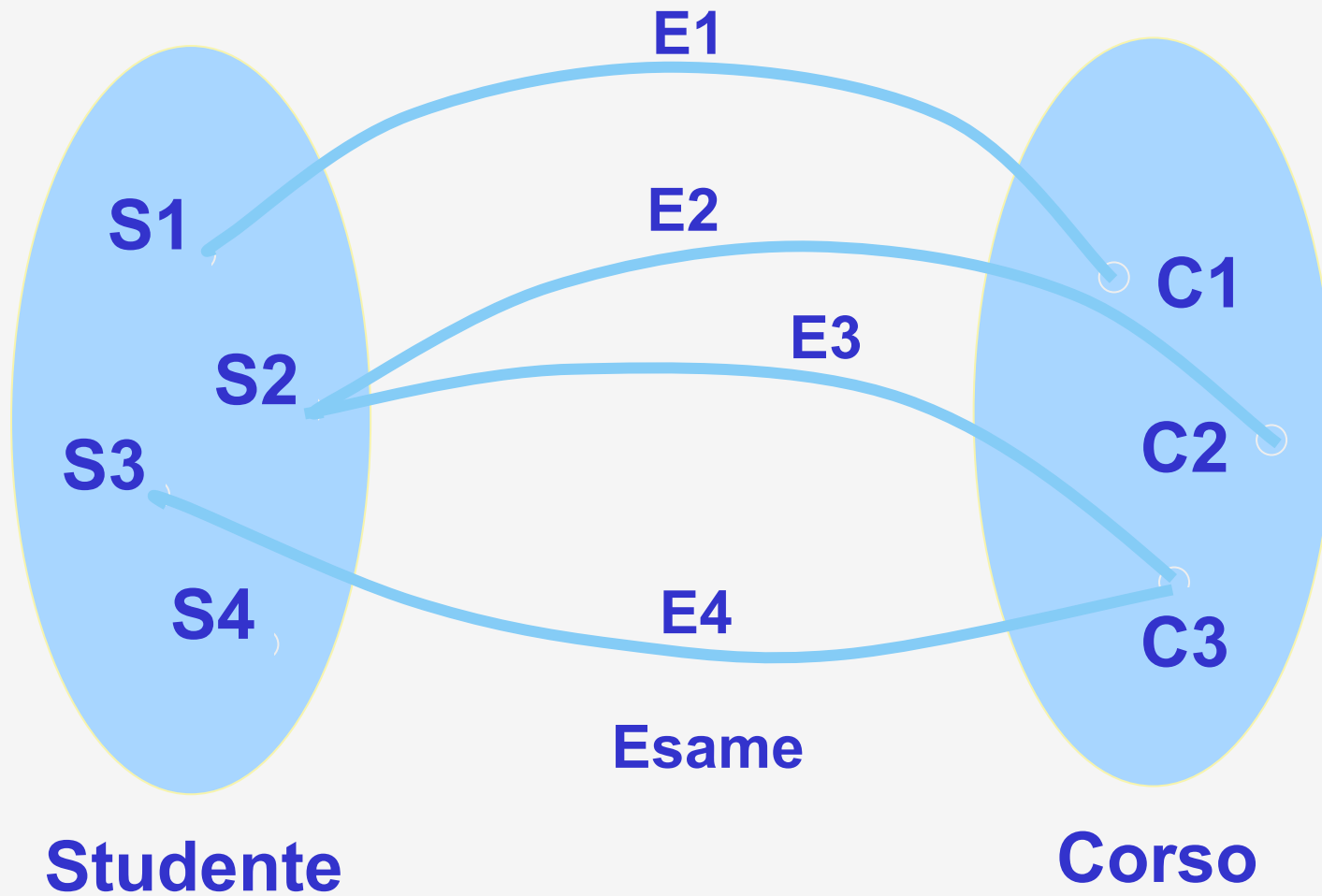
# Rappresentazione grafica di relationship



## Relationship, commenti

- Ogni relationship ha un nome che la identifica univocamente nello schema:
  - nomi espressivi
  - opportune convenzioni
    - singolare
    - sostantivi invece che verbi (se possibile)

# Esempi di occorrenze

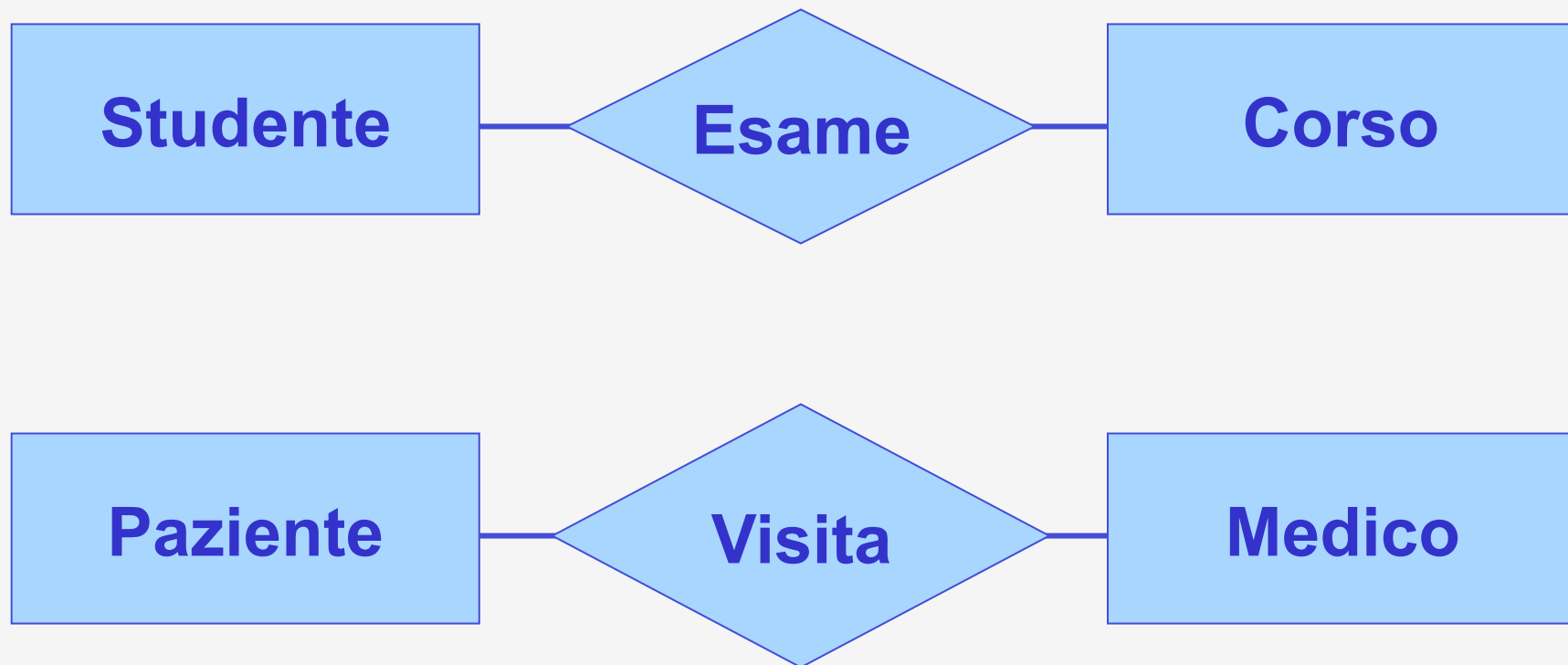


# Relationship, occorrenze

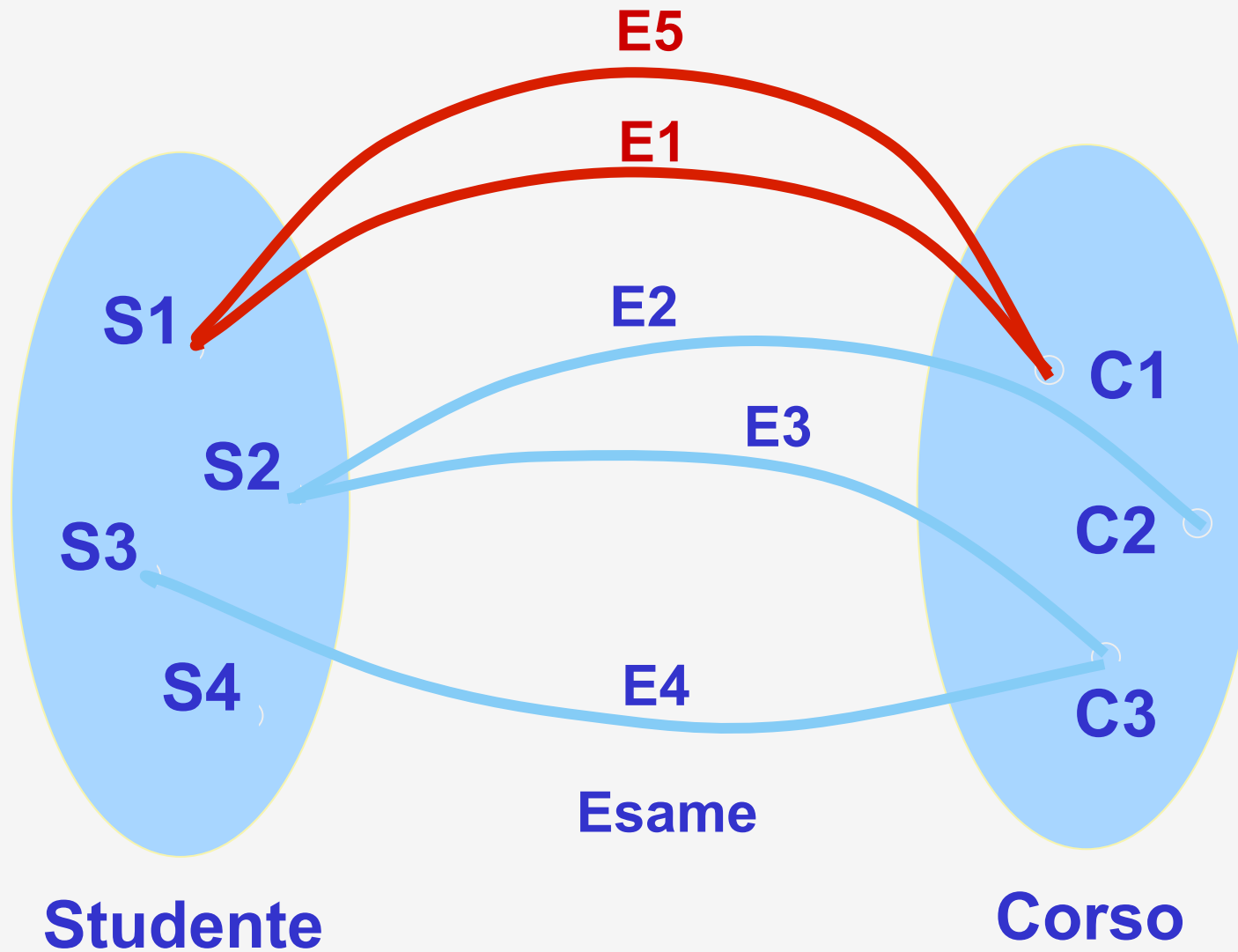
- Una occorrenza di una relationship binaria è coppia di occorrenze di entità, una per ciascuna entità coinvolta
- Una occorrenza di una relationship n-aria è una n-upla di occorrenze di entità, una per ciascuna entità coinvolta
- Nell'ambito di una relationship non ci possono essere occorrenze (coppie, ennuple) ripetute



## Relationship corrette?



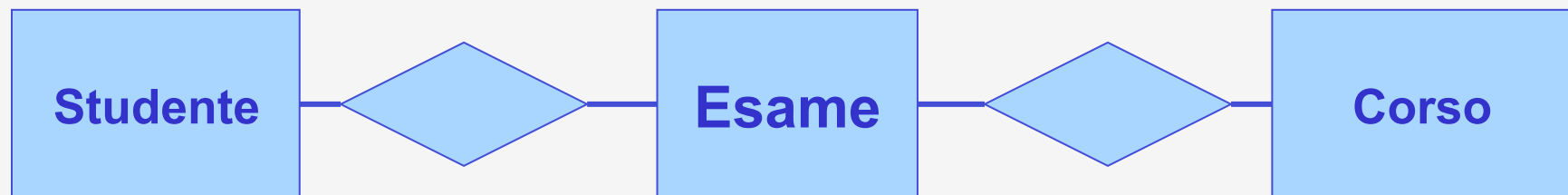
# Attenzione



## "Promuoviamo" la relationship

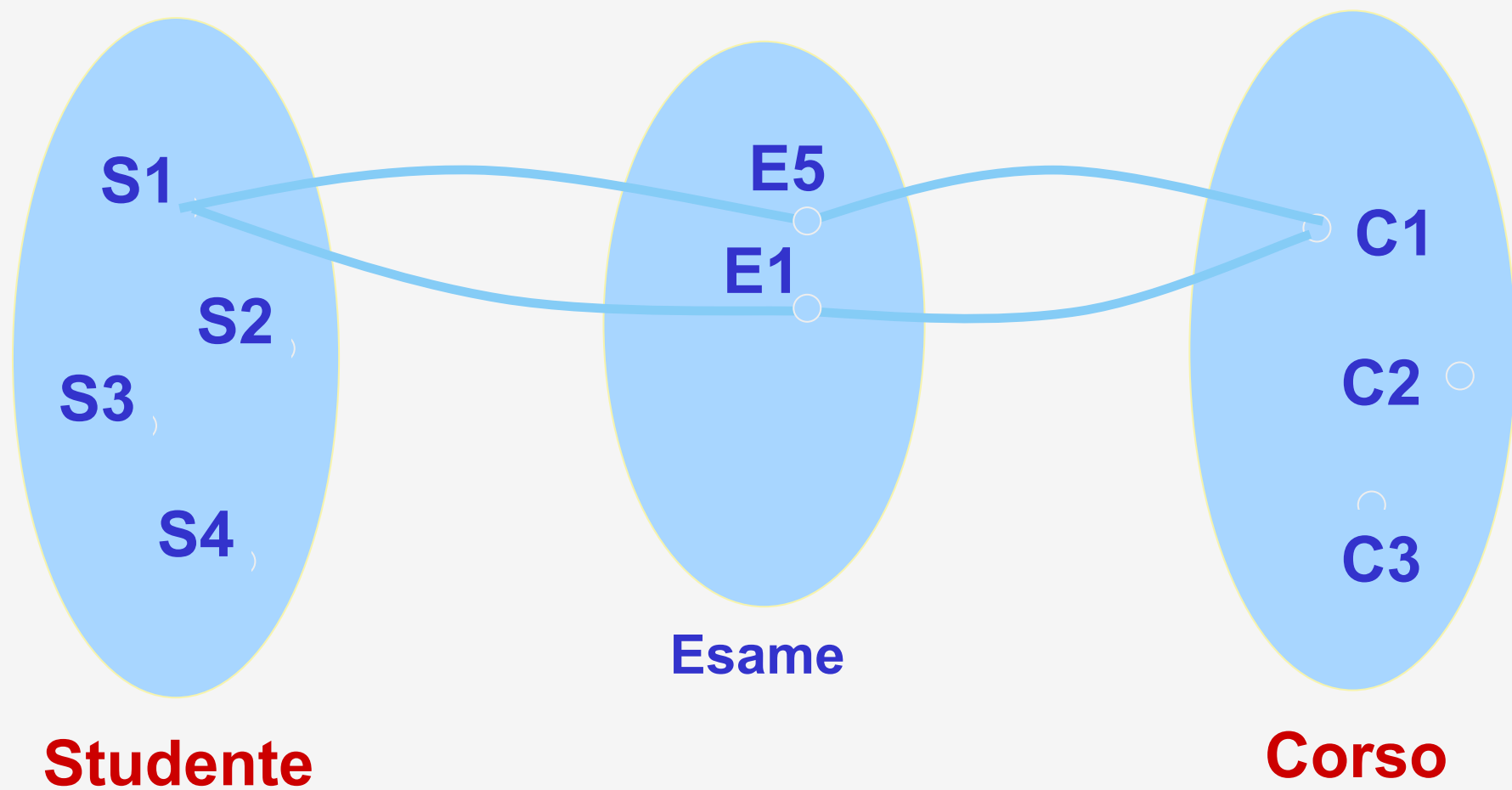


La relazione **Esame** non è in grado di descrivere che uno **Studente** abbia sostenuto più volte lo stesso esame relativo a quel **Corso**

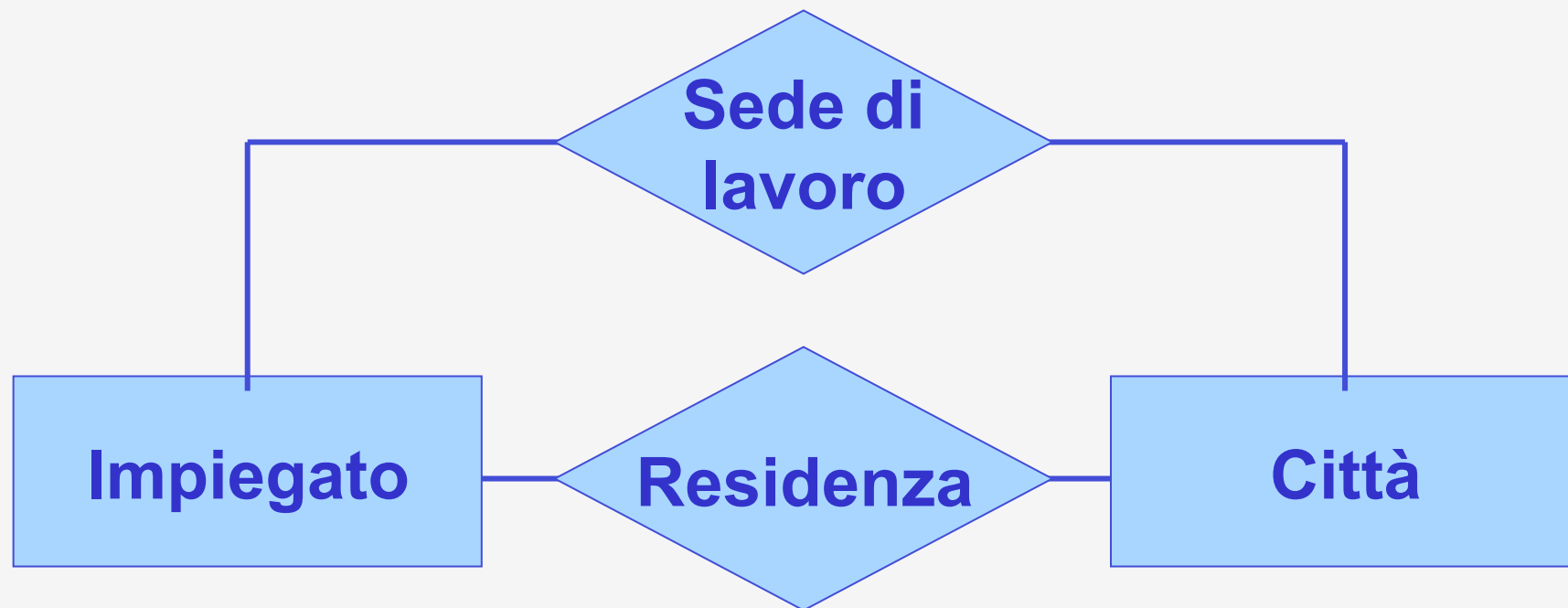


La soluzione sta nel rappresentare **Esame** come entità, collegata mediante relazioni a **Studente** e **Corso**

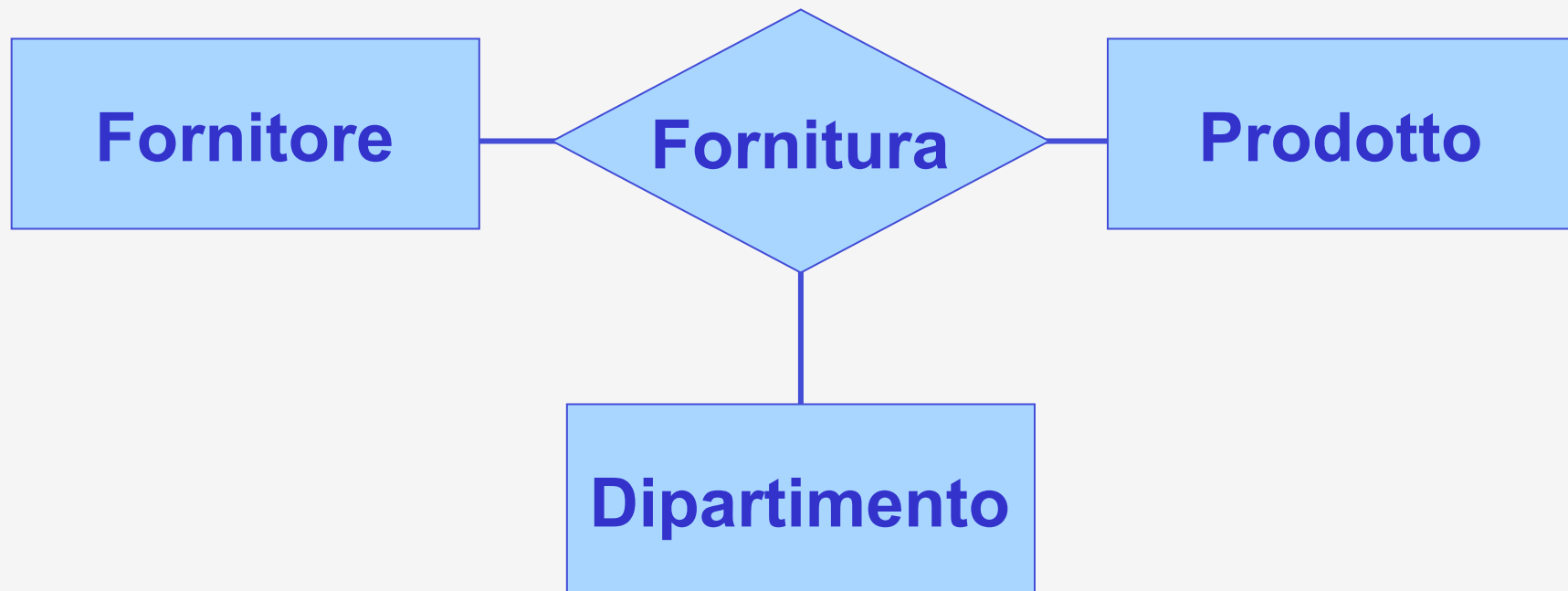
## Con l'entità Esame



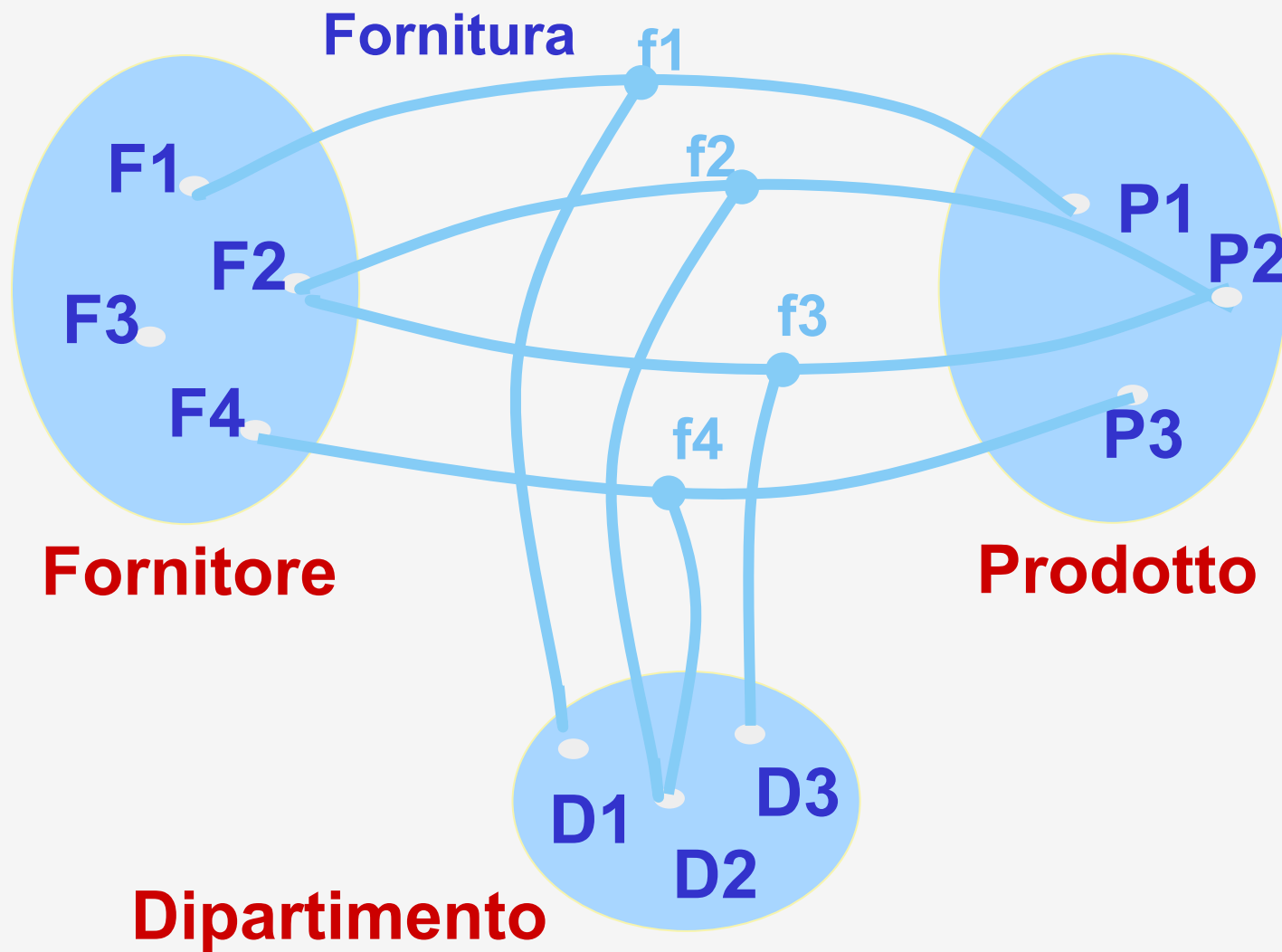
## Due relationship sulle stesse entità



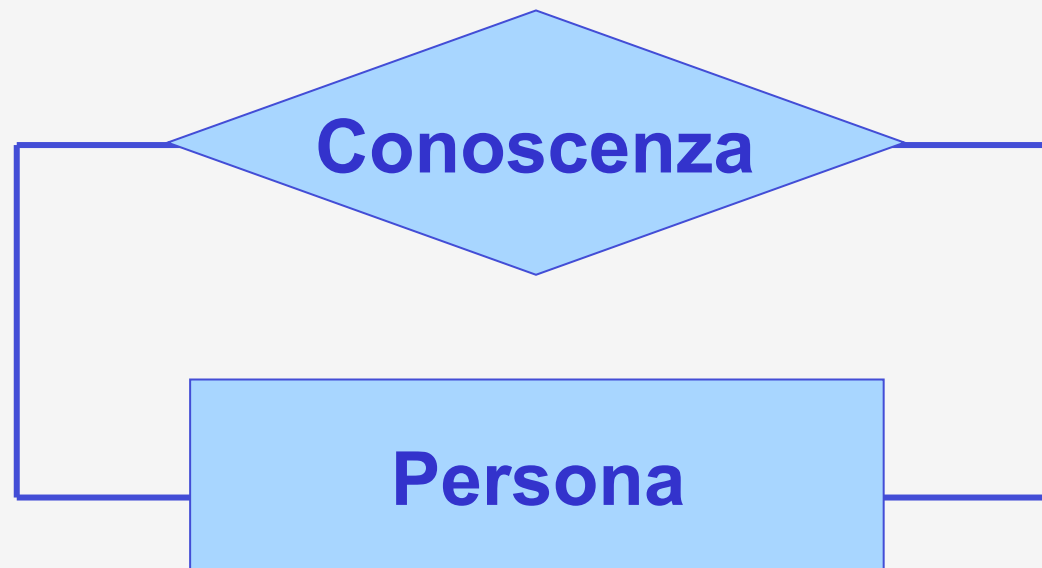
## Relationship n-aria



# Esempi di occorrenze

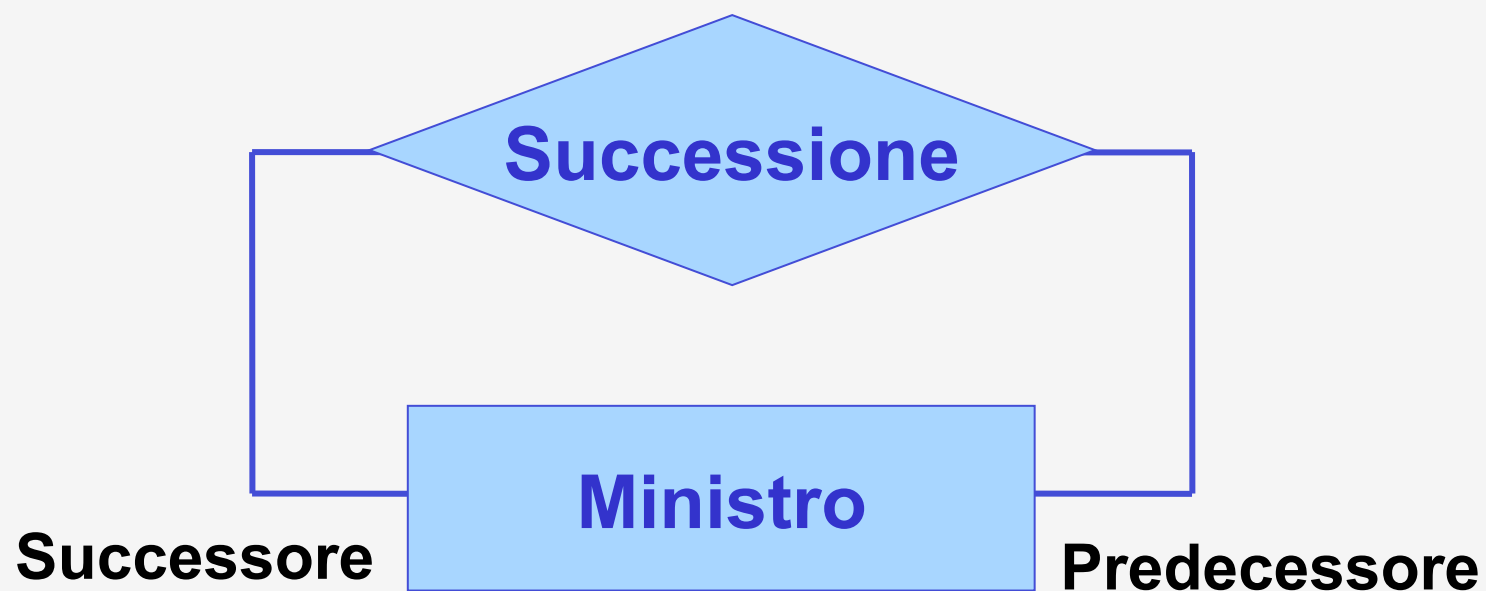


## **Relationship ricorsiva: coinvolge “due volte” la stessa entità**

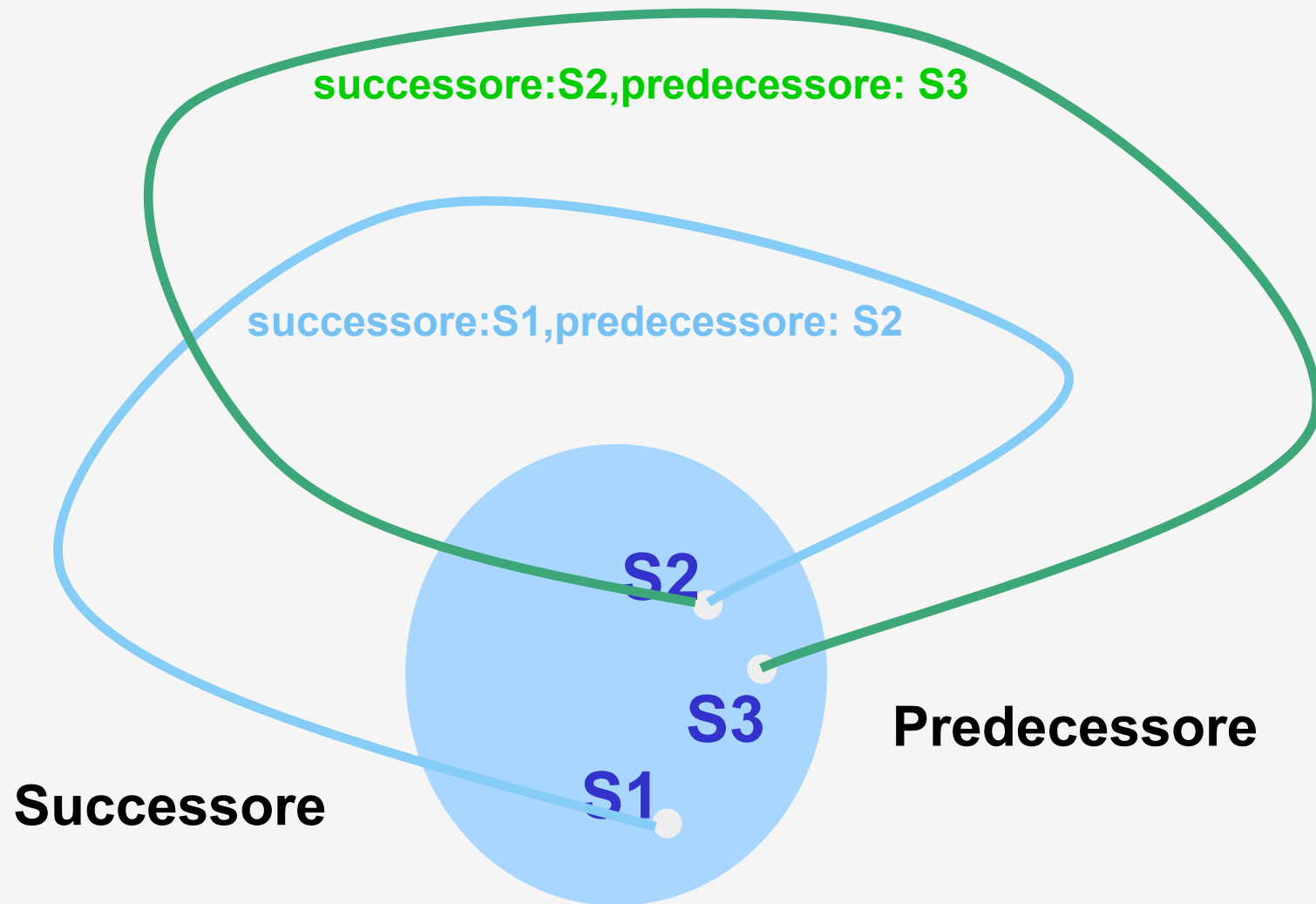




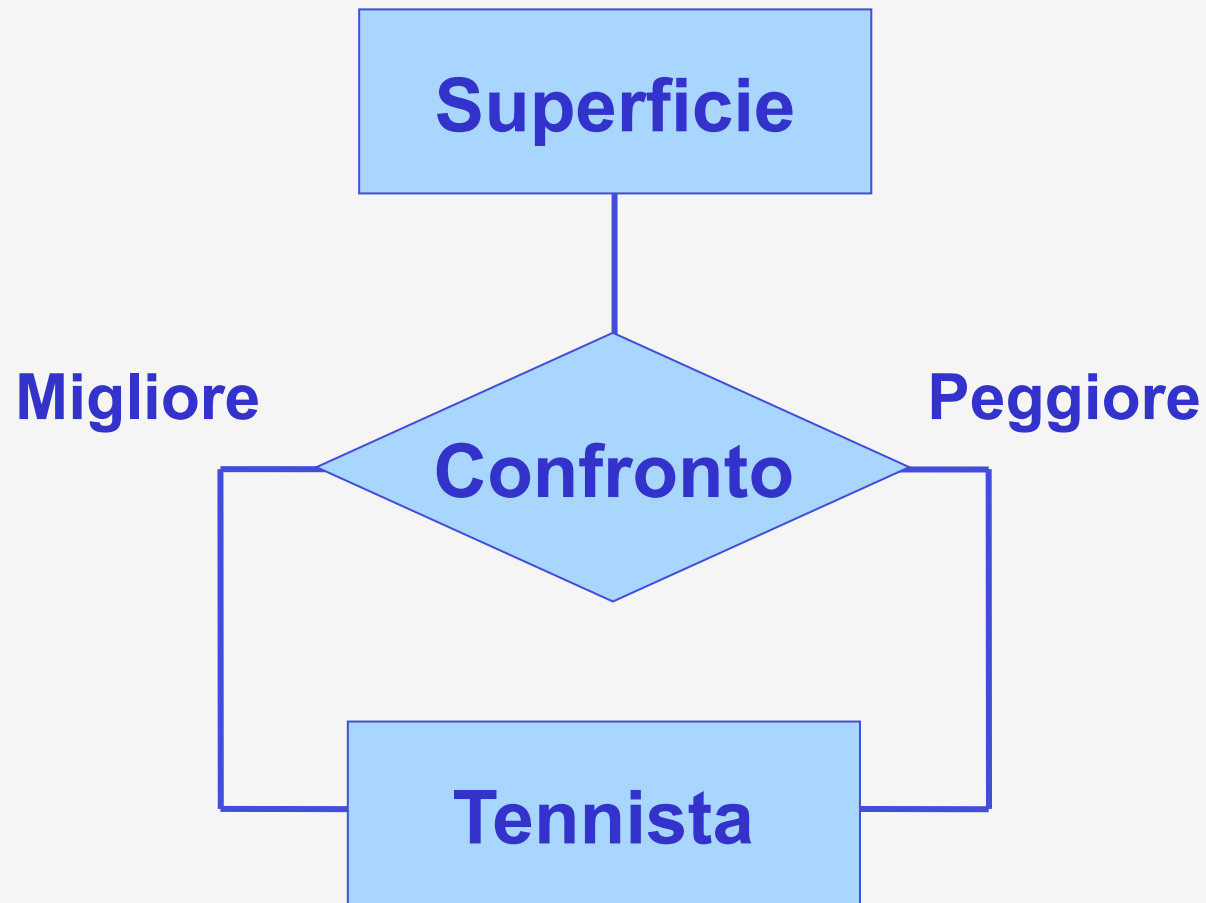
## Relationship ricorsiva con “ruoli”



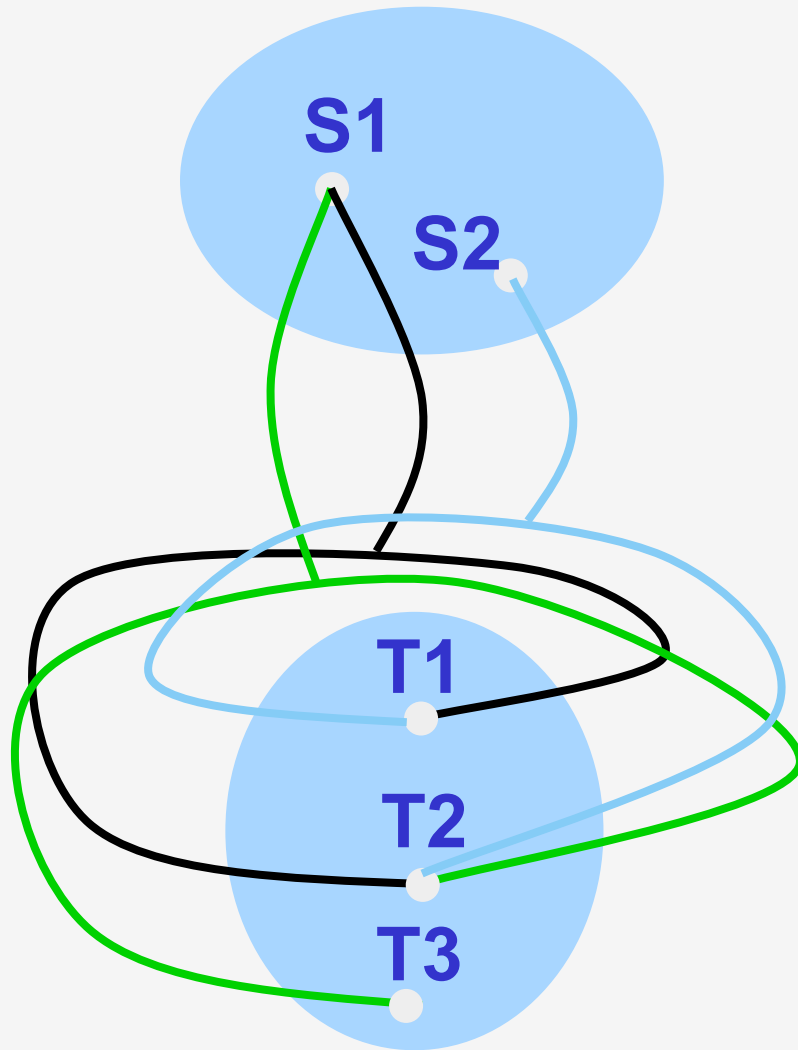
## Esempi di occorrenze (2)



# Relationship ternaria ricorsiva



## Esempi di occorrenze (3)



T1 è migliore di T2 su S2

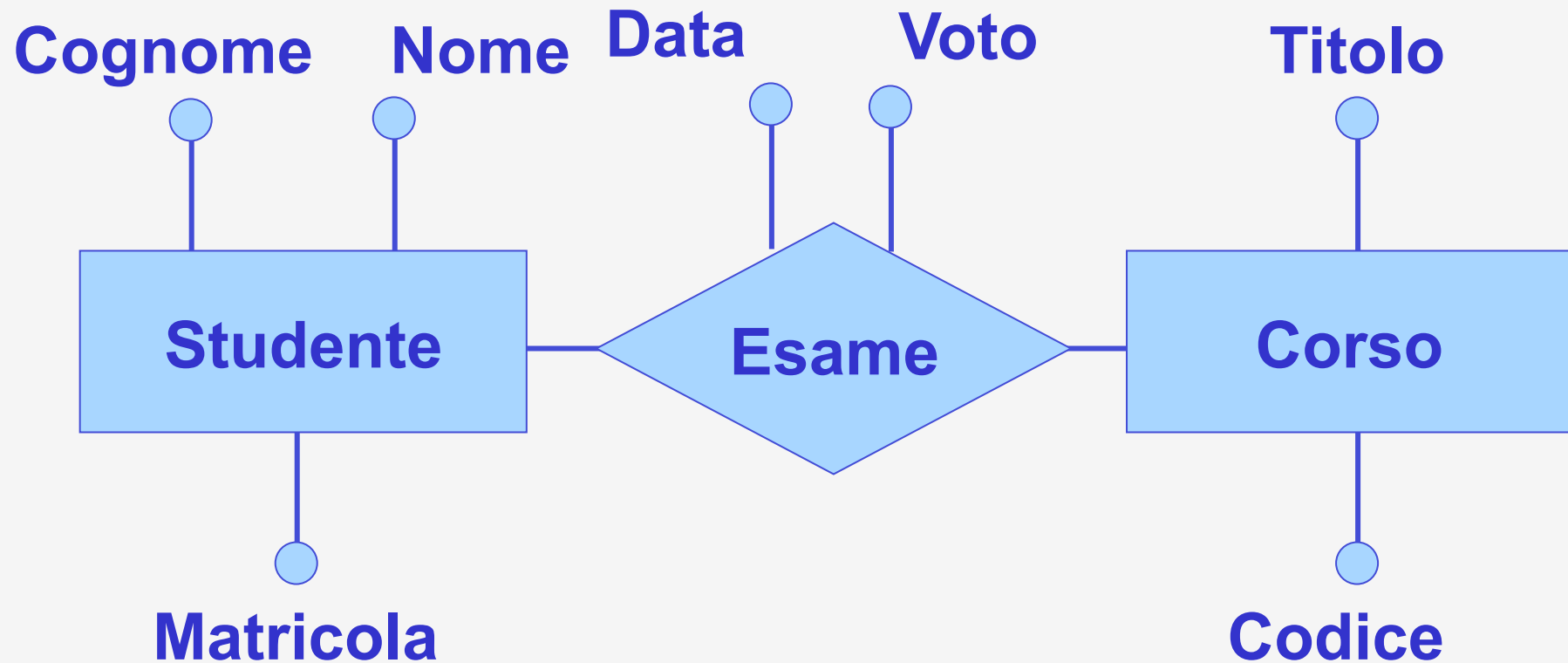
T2 è migliore di T1 su S1

T3 è migliore di T2 su S1

# Attributo

- Proprietà elementare di un' entità o di una relationship, di interesse ai fini dell' applicazione
- Associa ad ogni occorrenza di entità o relationship un valore appartenente a un insieme detto dominio dell' attributo

## Attributi, rappresentazione grafica



# Esempi di occorrenze (4)

Matricola: 34567

Cognome: Rossi

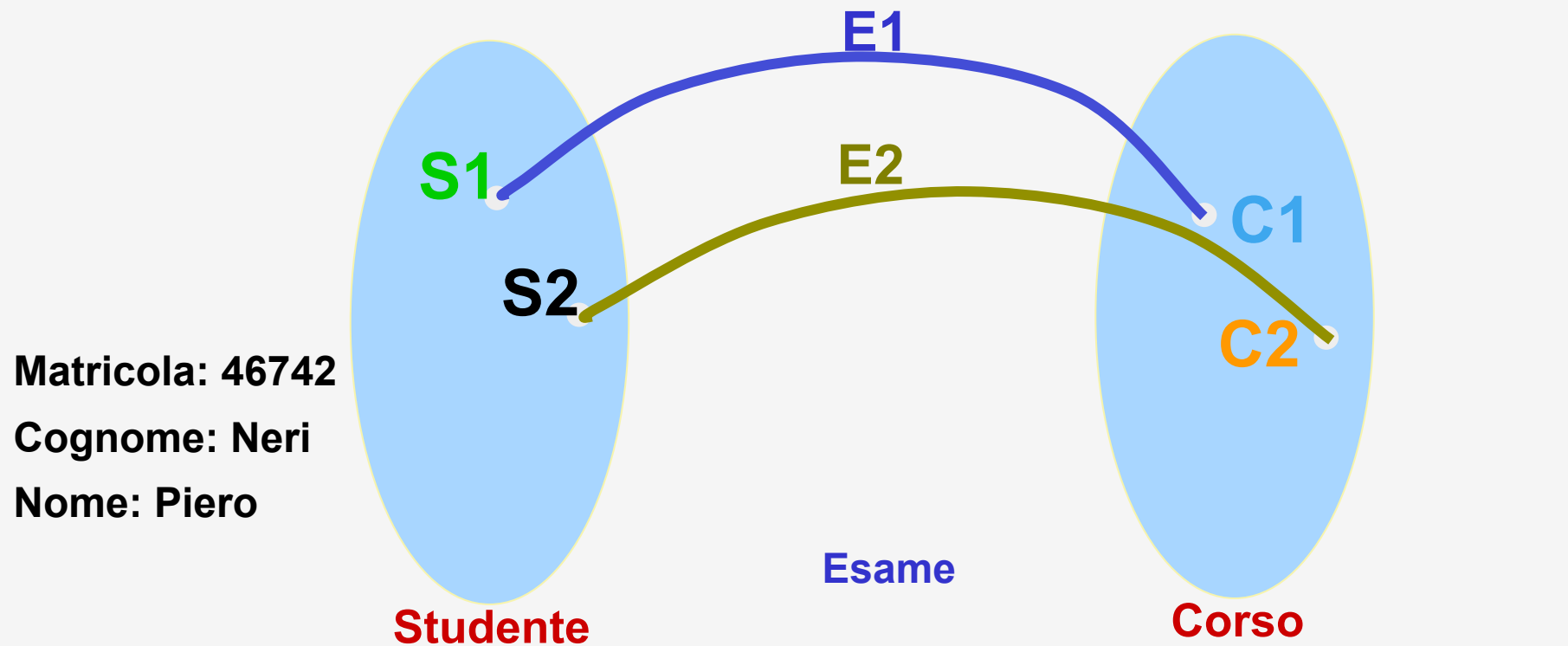
Nome: Mario

Data: 25/07/2004

Voto: 26

Codice: Inf205

Titolo: Basi di dati

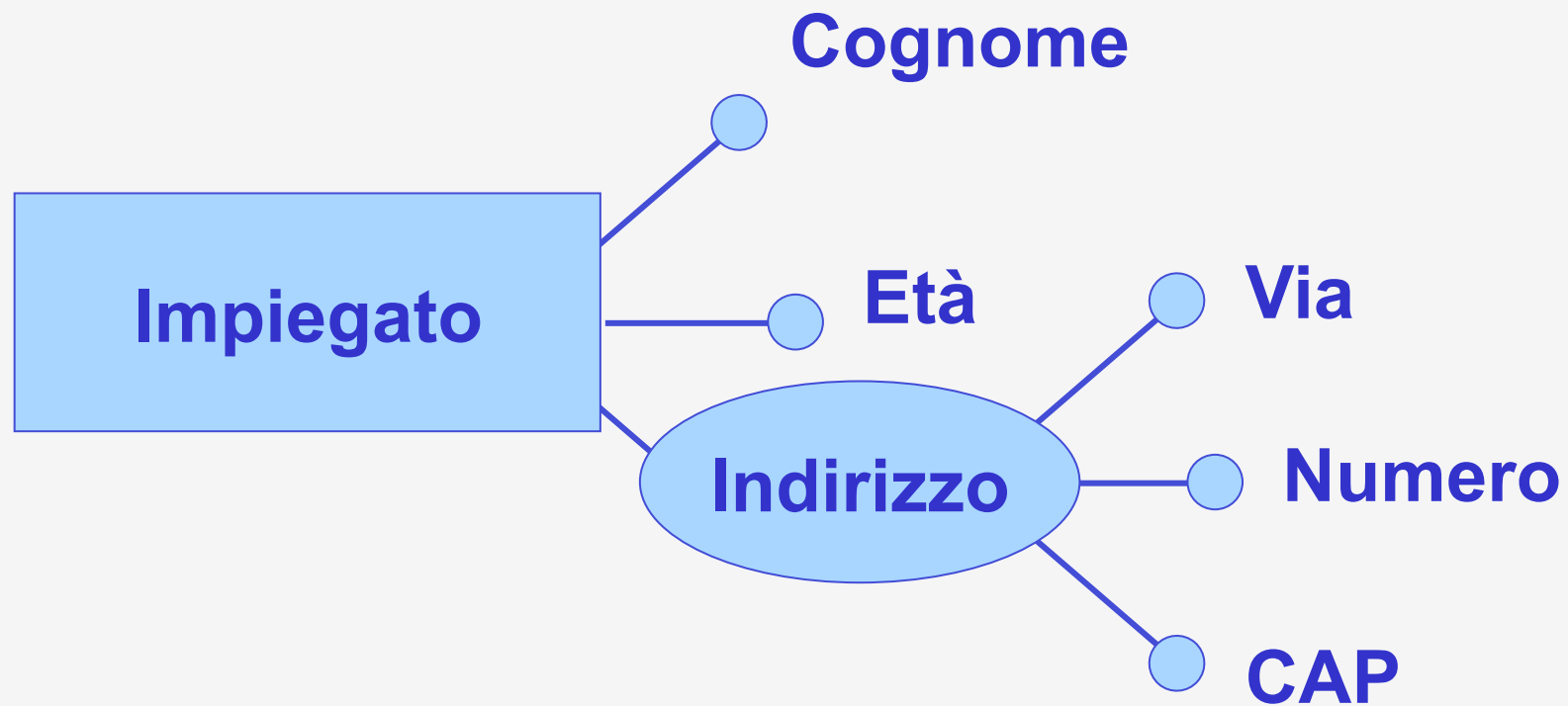


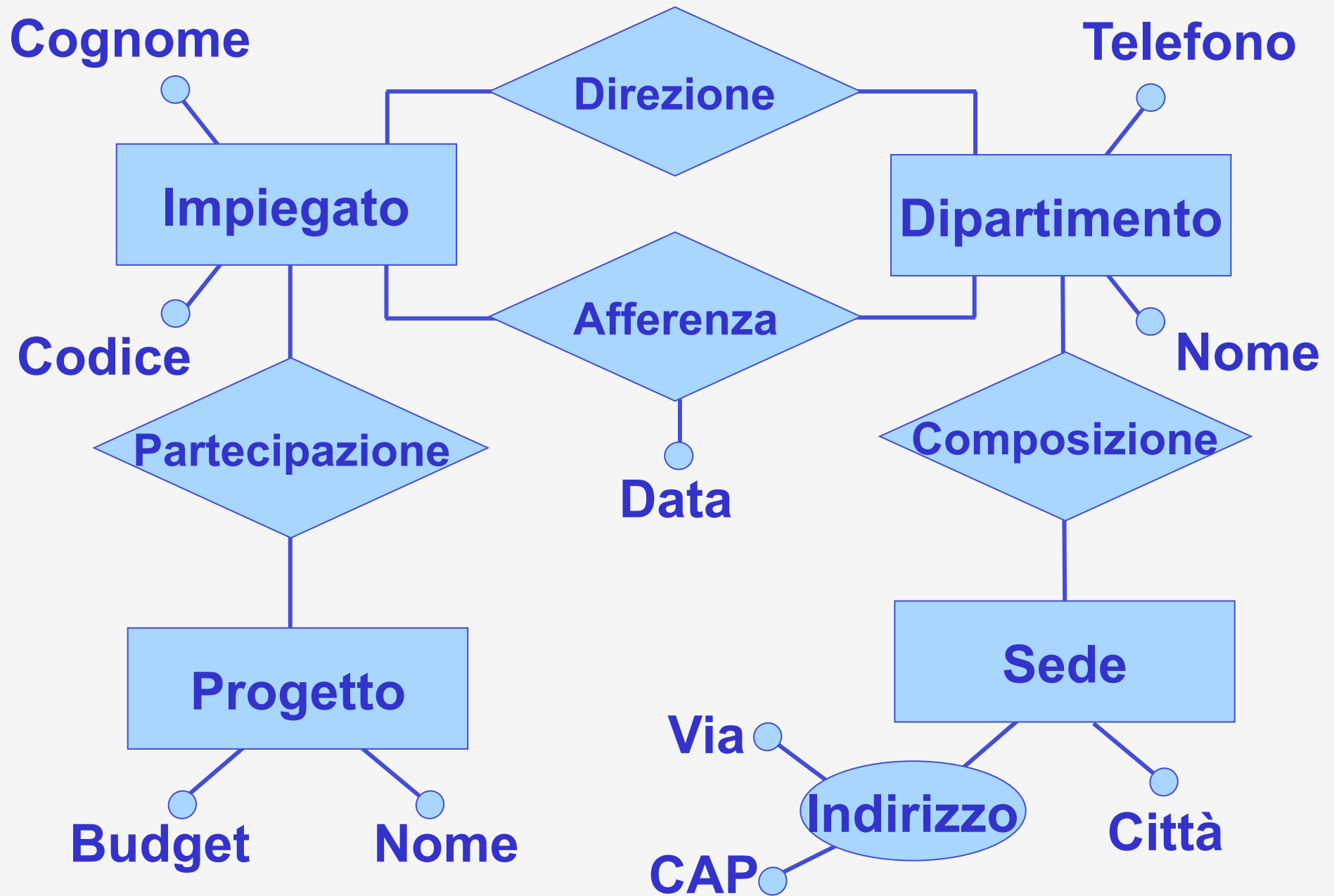
# Attributi composti

- Raggruppano attributi di una medesima entità o relationship che presentano affinità nel loro significato o uso
- Esempio:
  - Via, Numero civico e CAP formano un Indirizzo



# Rappresentazione grafica





## Altri costrutti del modello E-R

- Cardinalità
  - di relationship
  - di attributo
- Identificatore
  - interno
  - esterno
- Generalizzazione

## Cardinalità di relationship

- 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 una entità può partecipare

## Esempio di cardinalità

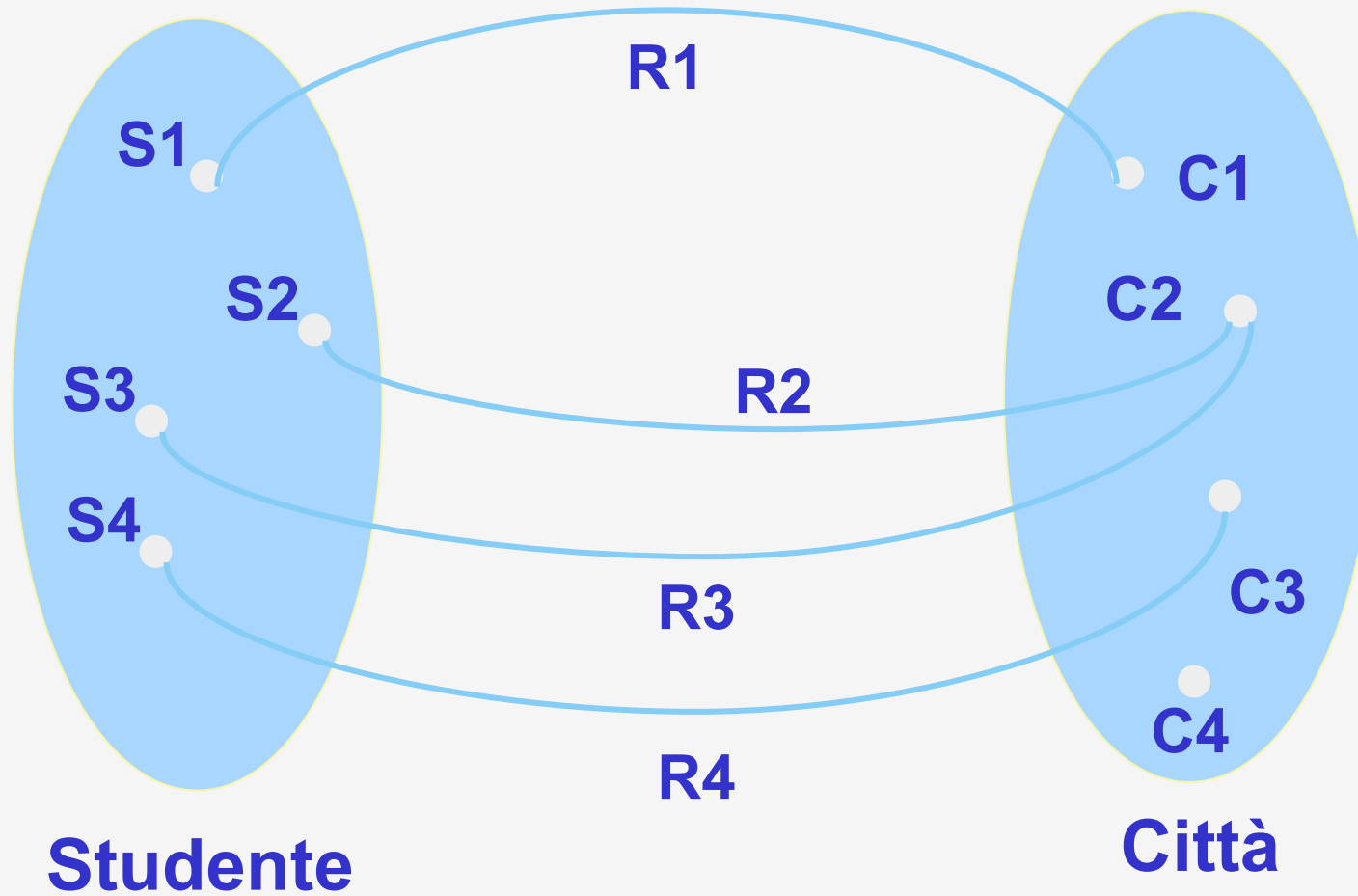


- Ad un **Impiegato** deve essere assegnato almeno **1** incarico ma non più di **5**
- Ogni **Incarico** può non essere assegnato a nessuno (**0**) oppure può essere assegnato a un numero inferiore o uguale a **50** impiegati

# Cardinalità

- per semplicità usiamo solo tre simboli:
- 0 e 1 per la cardinalità minima:
  - 0 = “partecipazione **opzionale**”
  - 1 = “partecipazione **obbligatoria**”
- 1 e “N” per la massima:
  - “N” non pone alcun limite

# Occorrenze di Residenza



# Cardinalità di Residenza

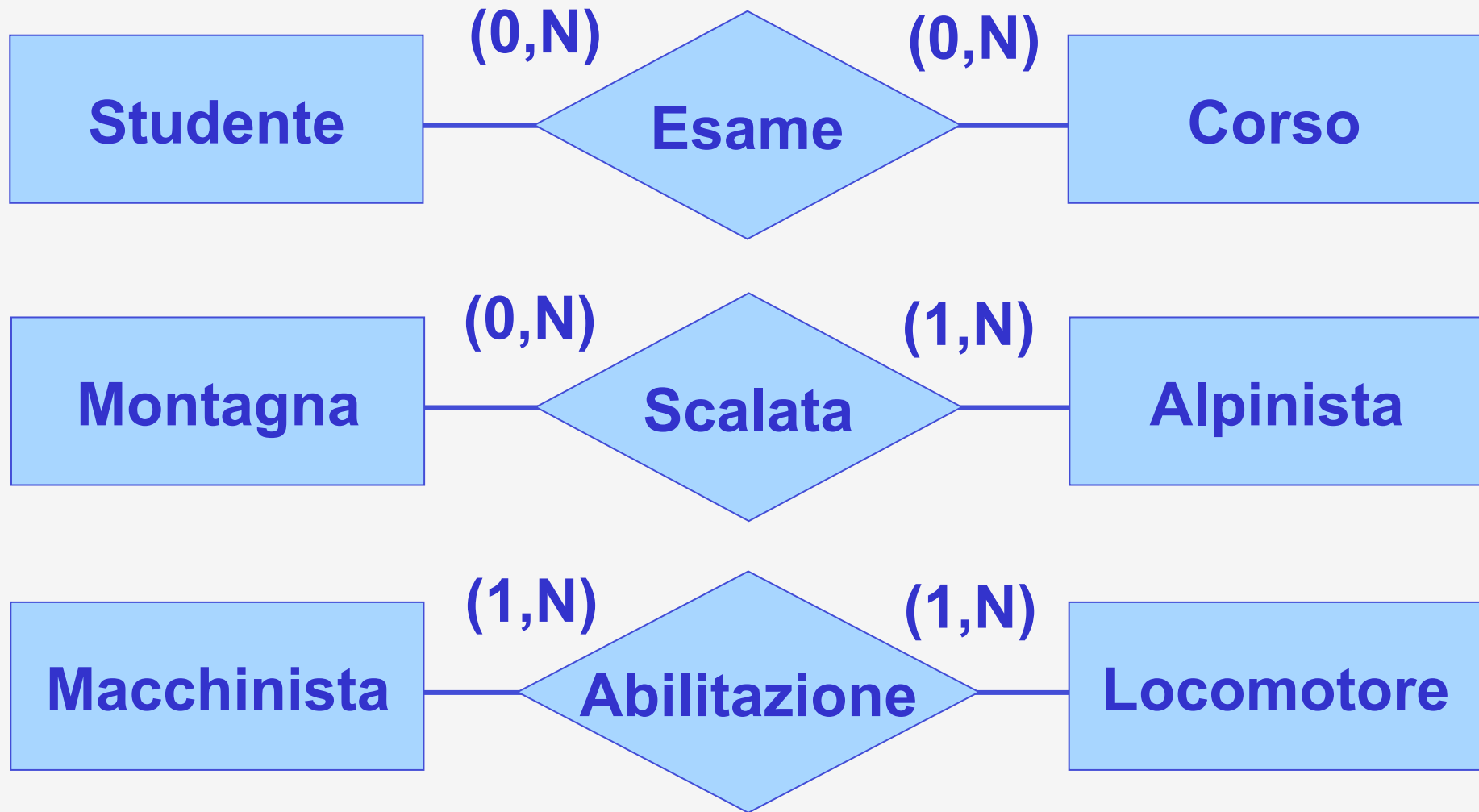




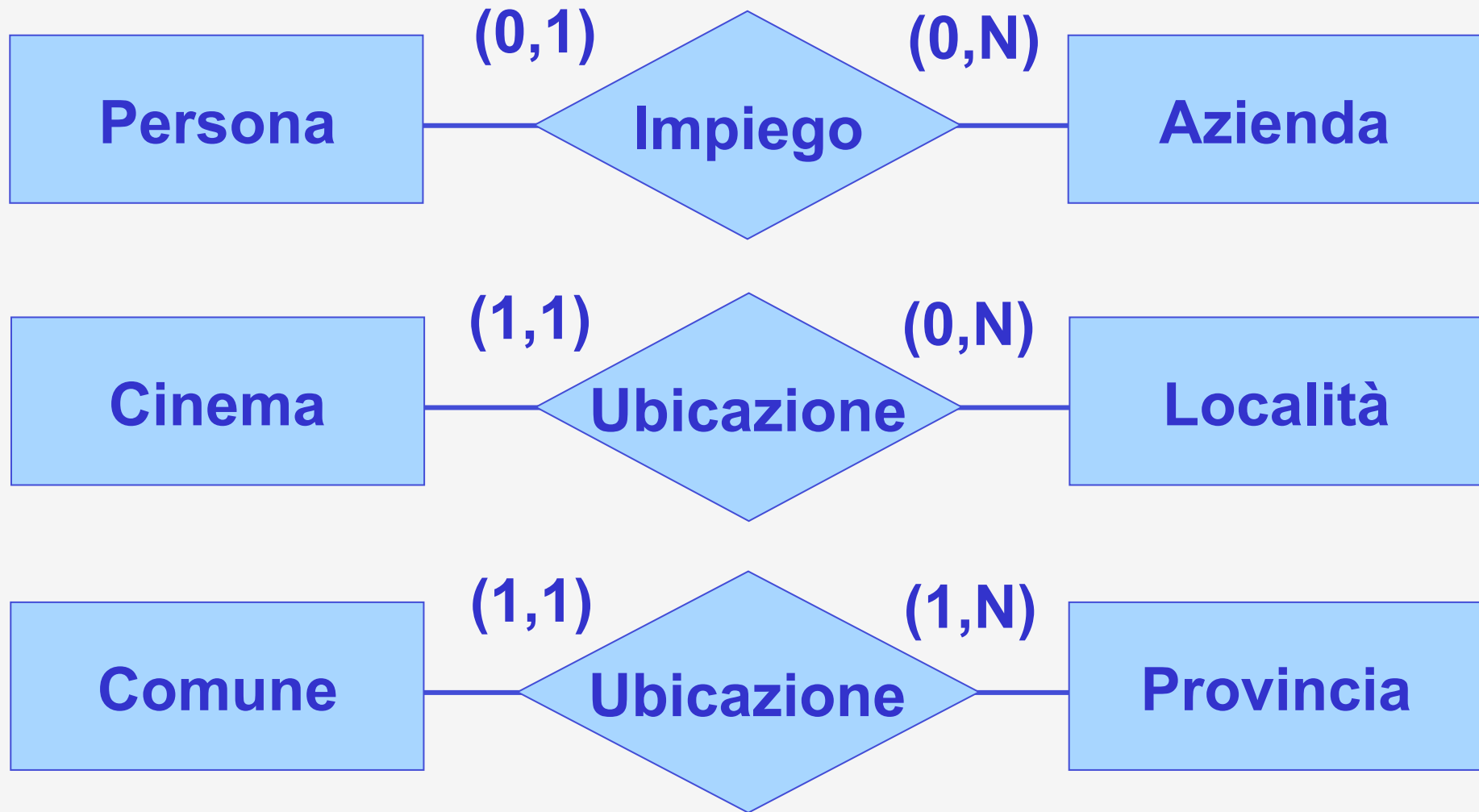
## Tipi di relationship

- Con riferimento alle cardinalità **massime**, abbiamo relationship:
  - uno a uno
  - uno a molti
  - molti a molti

## Relationship “molti a molti”



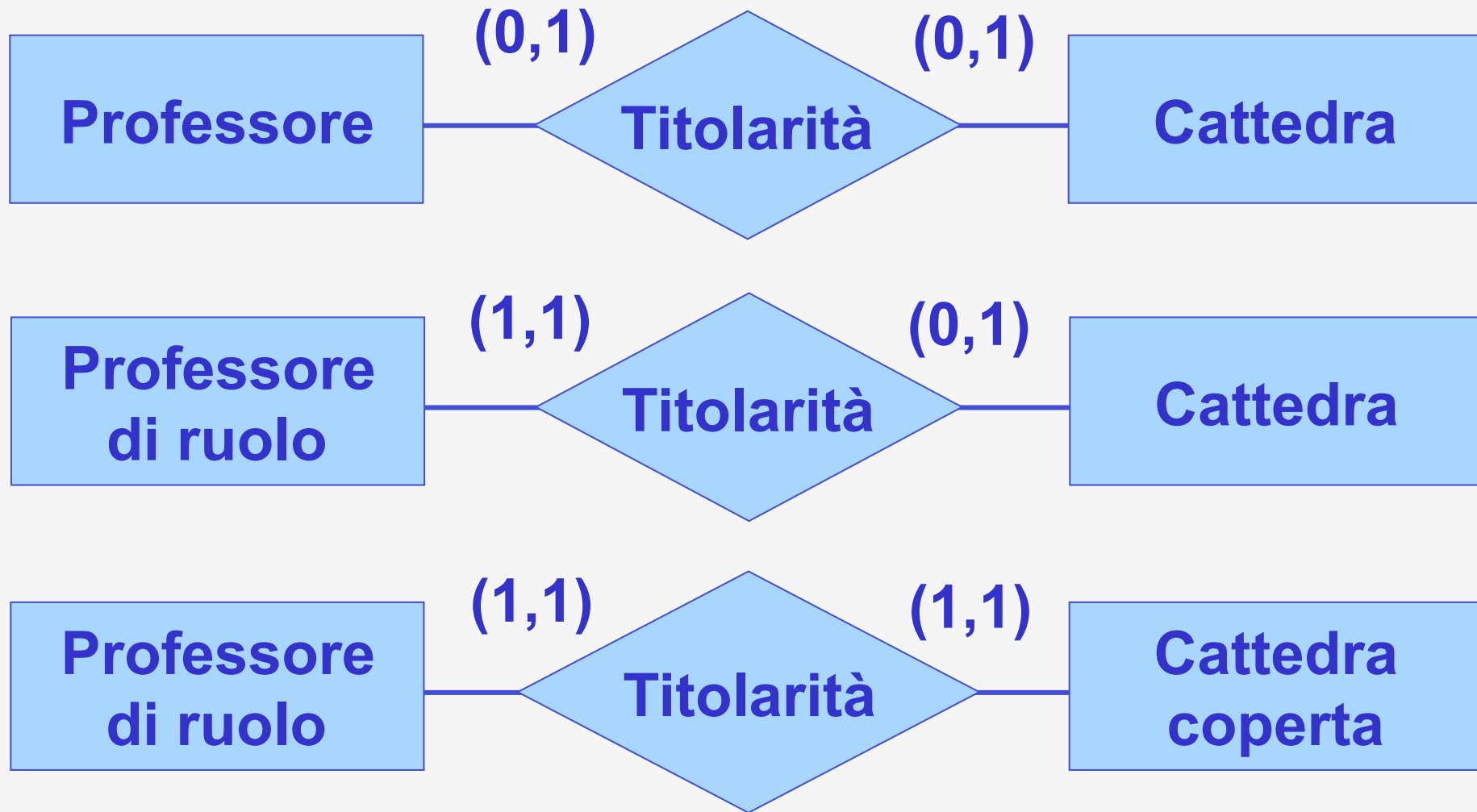
## Relationship “uno a molti”



## Due avvertenze

- Attenzione al "verso" nelle relationship uno a molti
- le relationship obbligatorie-obbligatorie sono molto rare

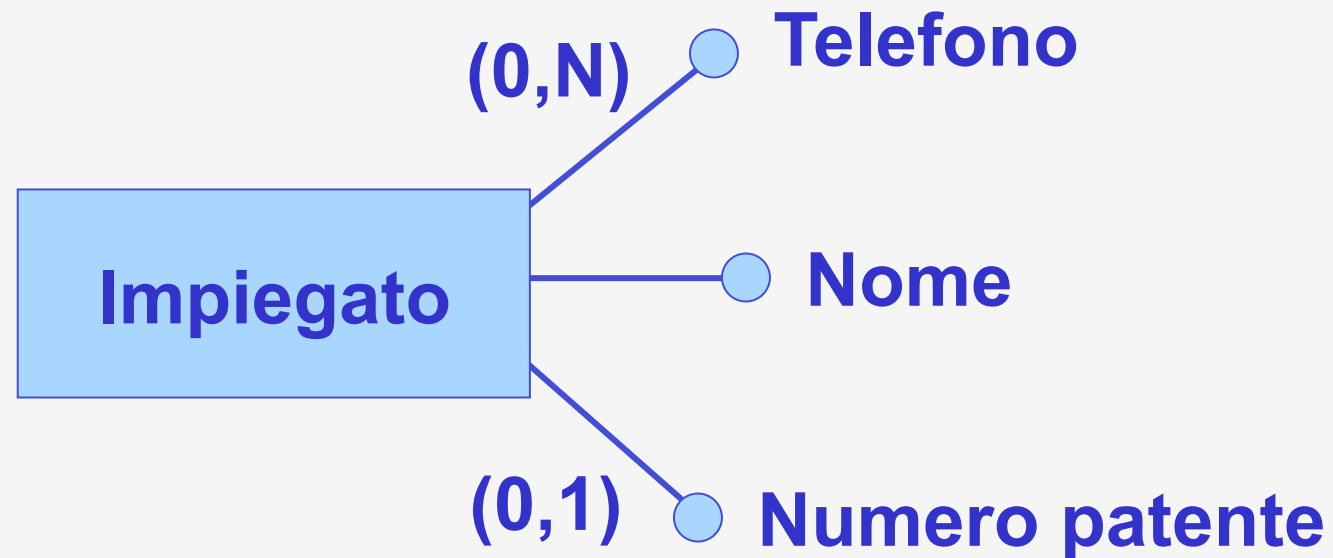
## Relationship “uno a uno”



## Cardinalità di attributi

- E' possibile associare delle cardinalità anche agli attributi, con due scopi:
  - indicare opzionalità ("informazione incompleta")
  - indicare attributi multivalore

# Rappresentazione grafica

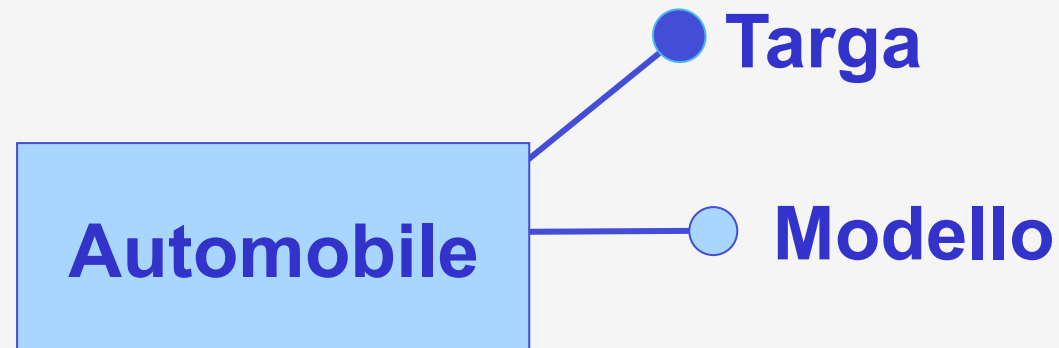


# Identificatore di una entità

- “strumento” per l’ identificazione univoca delle occorrenze di un’ entità
- costituito da:
  - attributi dell’ entità
    - identificatore interno
  - (attributi +) entità esterne attraverso relationship
    - identificatore esterno

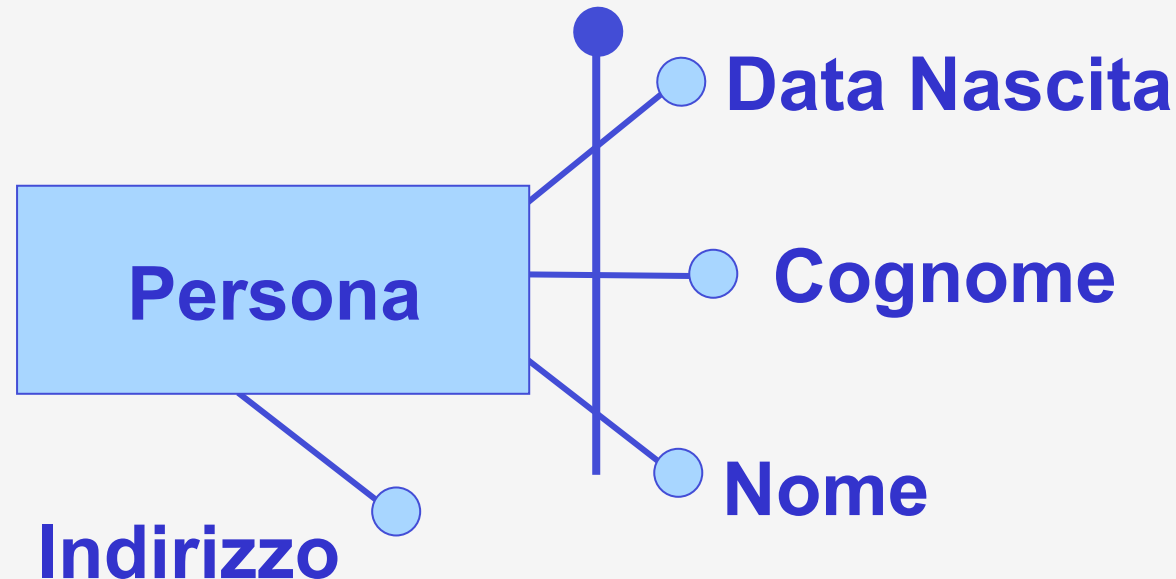


## Identificatori interni (1)



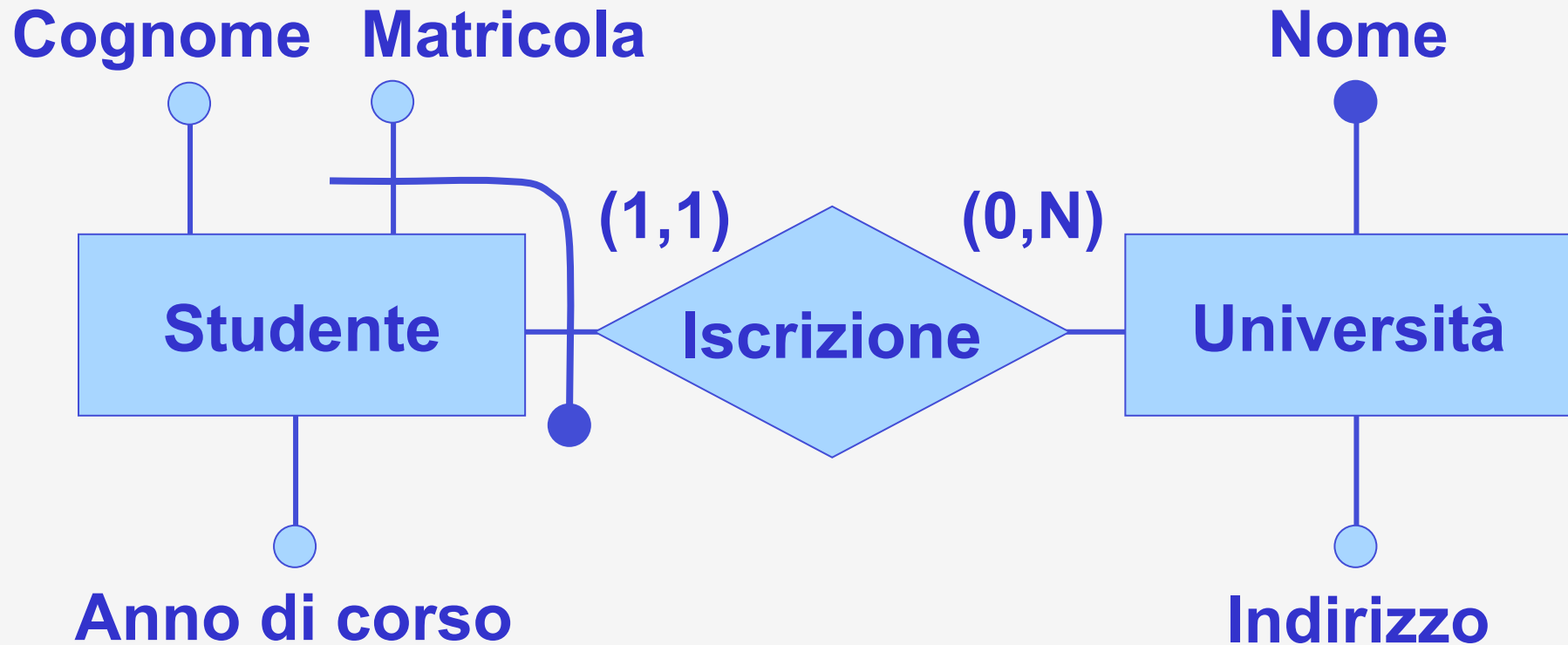
L'attributo **Targa** è **identificatore interno** dell'entità **Automobile** in quanto non posso esistere due automobili con la stessa targa

## Identificatori interni (2)



L'insieme degli attributi: **Nome**, **Cognome** e **Data di nascita** costituiscono **l'identificatore interno** dell'entità **Persona** in quanto non possono esistere (nel nostro caso) due persone aventi nome, cognome e data di nascita identici

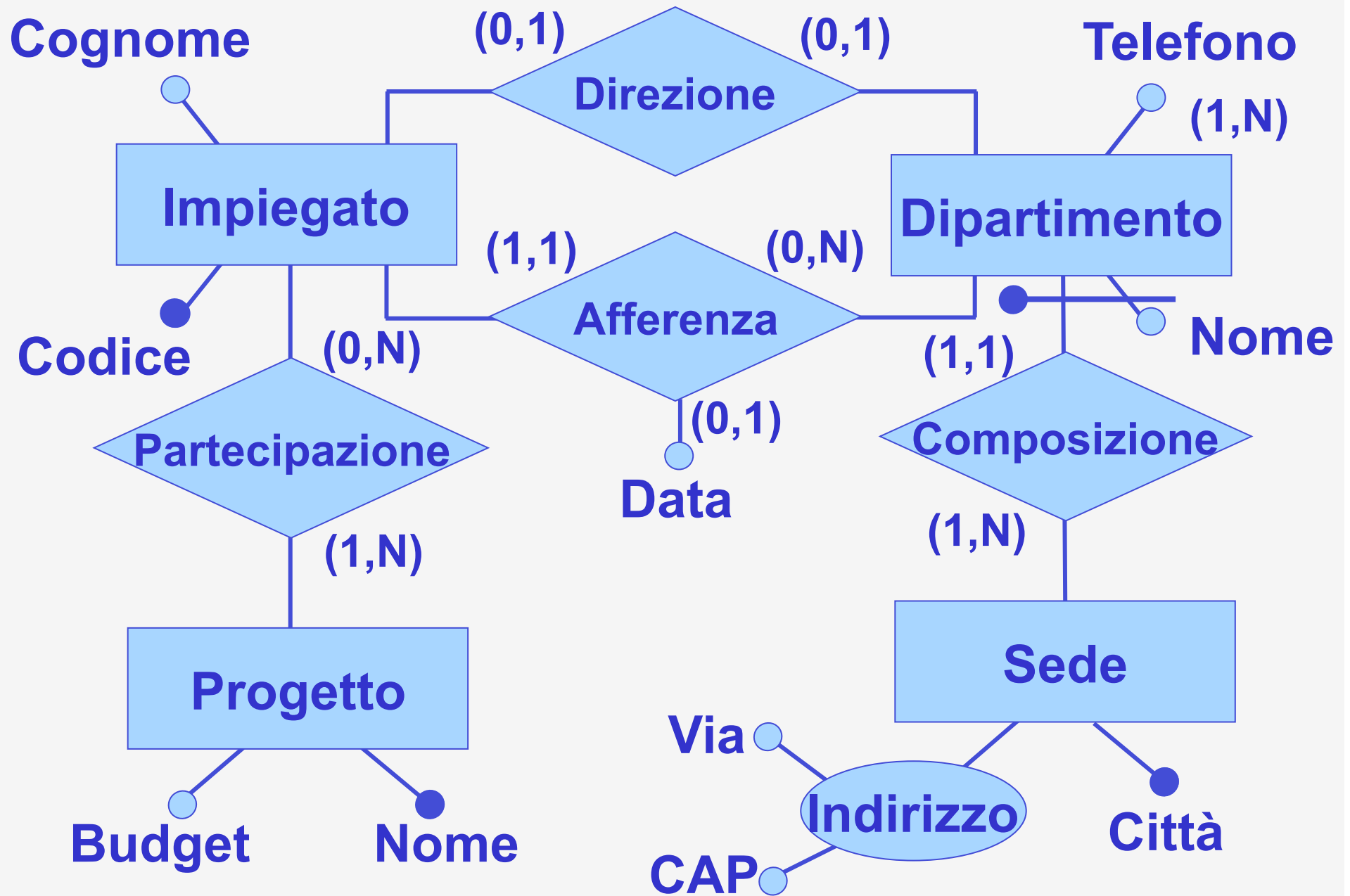
## Identificatore esterno



L'attributo **Matricola** e l'entità **Università** rappresentano l'**identificatore esterno** in quanto possiamo avere studenti con lo stesso numero di matricola, ma appartenenti ad università diverse

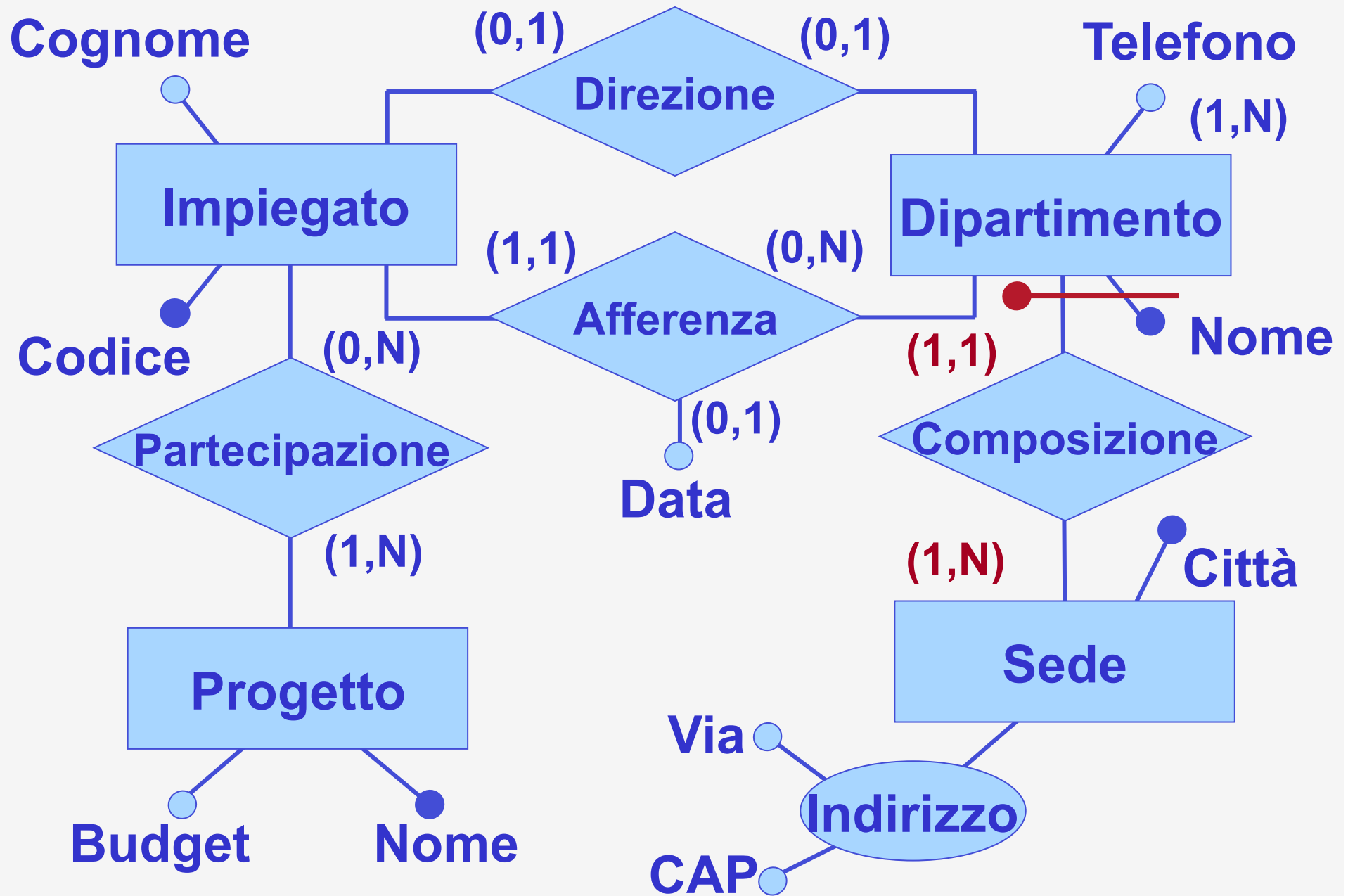
## Alcune osservazioni

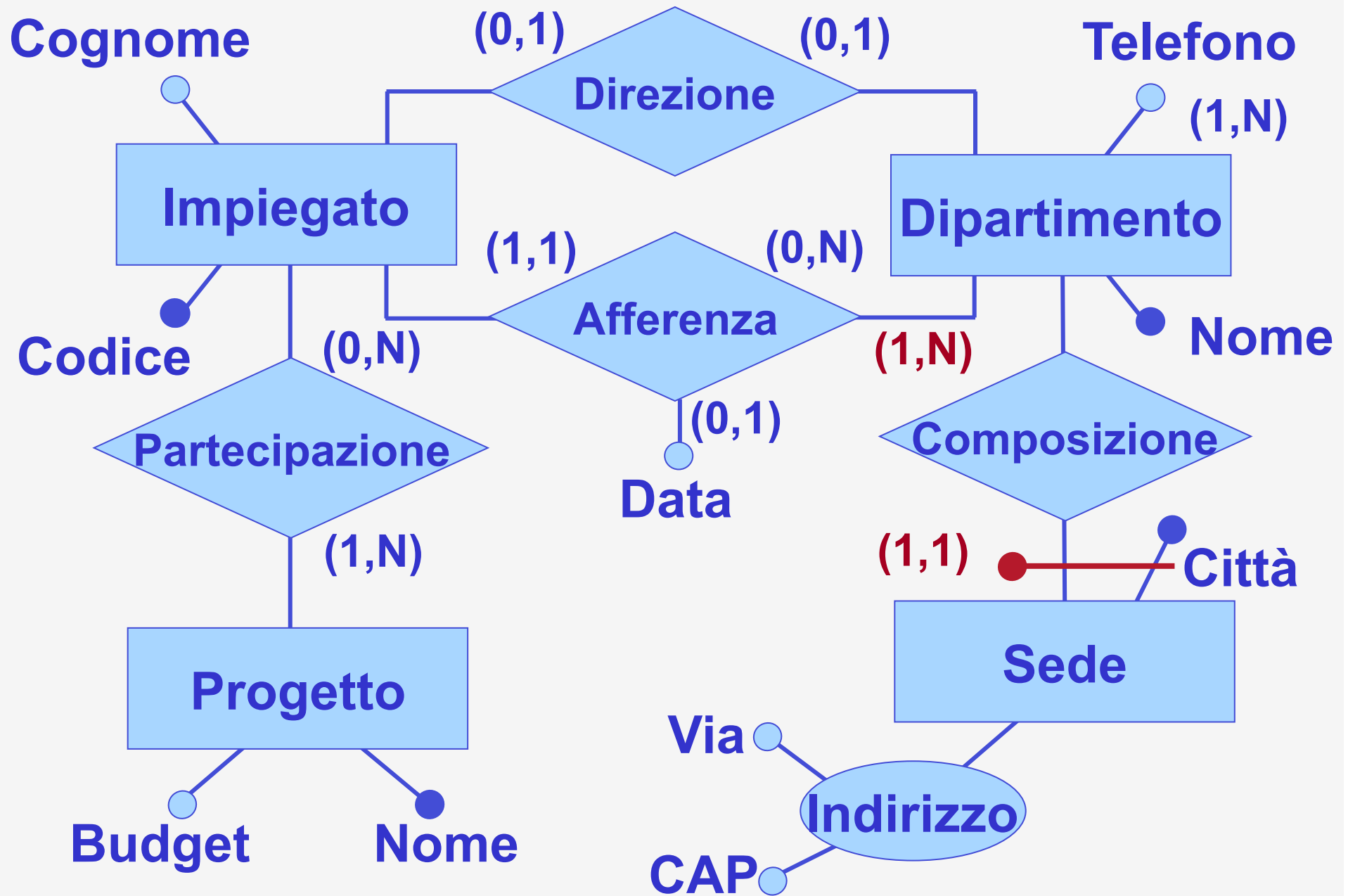
- ogni entità deve possedere almeno un identificatore, ma può averne in generale più di uno
- una identificazione esterna è possibile solo attraverso una relationship a cui l'entità da identificare partecipa con cardinalità (1,1)
- **Perché non parliamo degli identificatori delle relationship?**
  - perché è opportuno privilegiare l'inserimento di attributi nelle entità piuttosto che nelle relationship



## Attenzione

- Differenze apparentemente piccole in cardinalità e identificatori possono cambiare di molto il significato ...







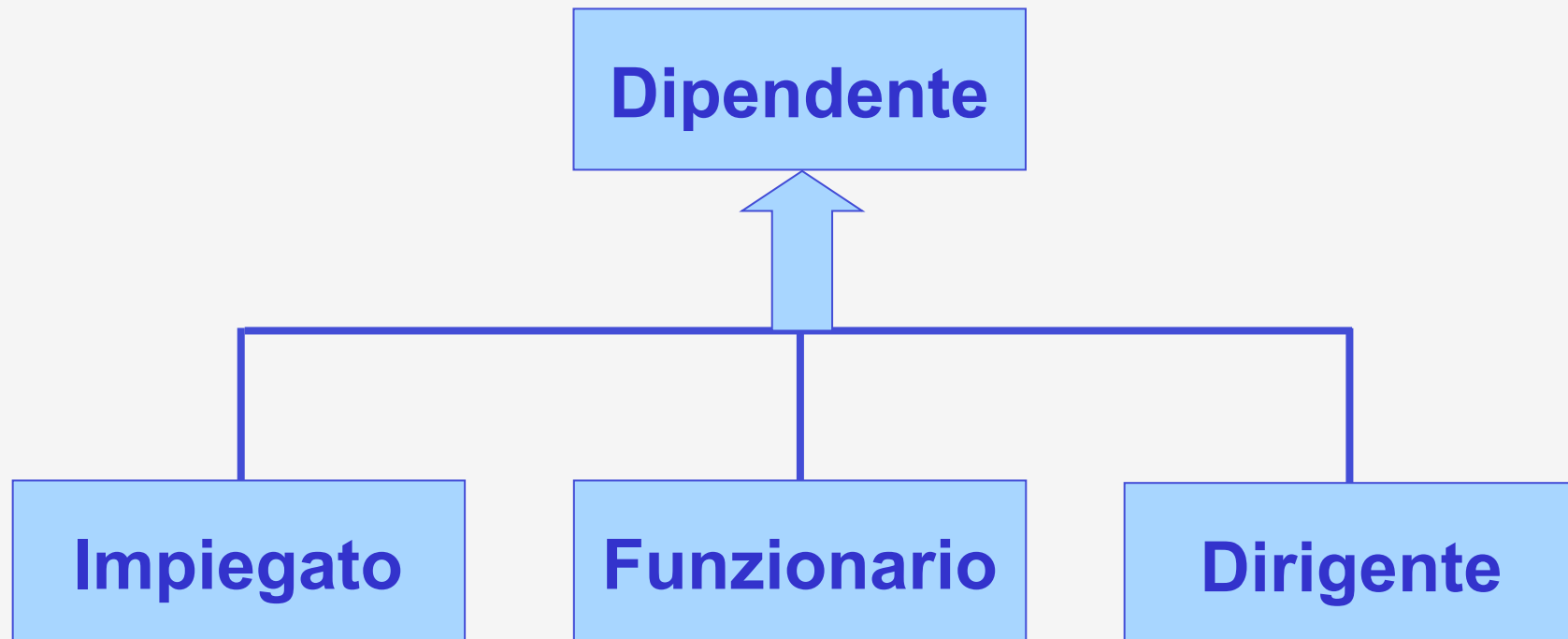
## Identificatori esterni e cardinalità

- Nel primo schema E-R (slide 71) viene assegnato ad ogni **Dipartimento** una ed una sola **Sede**: possiamo avere quindi anche dipartimenti con lo stesso **Nome**, l'importante è che la **Città** dove ha sede il dipartimento sia diversa
- Viceversa nel secondo schema E-R (slide 72) viene assegnato ad ogni **Sede** uno ed un solo **Dipartimento**: quindi possiamo avere nella stessa **Città** più dipartimenti ma con **Nome** diverso

# Generalizzazione

- mette in relazione una o più entità  $E_1, E_2, \dots, E_n$  con una entità  $E$ , che le comprende come casi particolari
  - $E$  è **generalizzazione** di  $E_1, E_2, \dots, E_n$
  - $E_1, E_2, \dots, E_n$  sono **specializzazioni** (o sottotipi) di  $E$

# Rappresentazione grafica

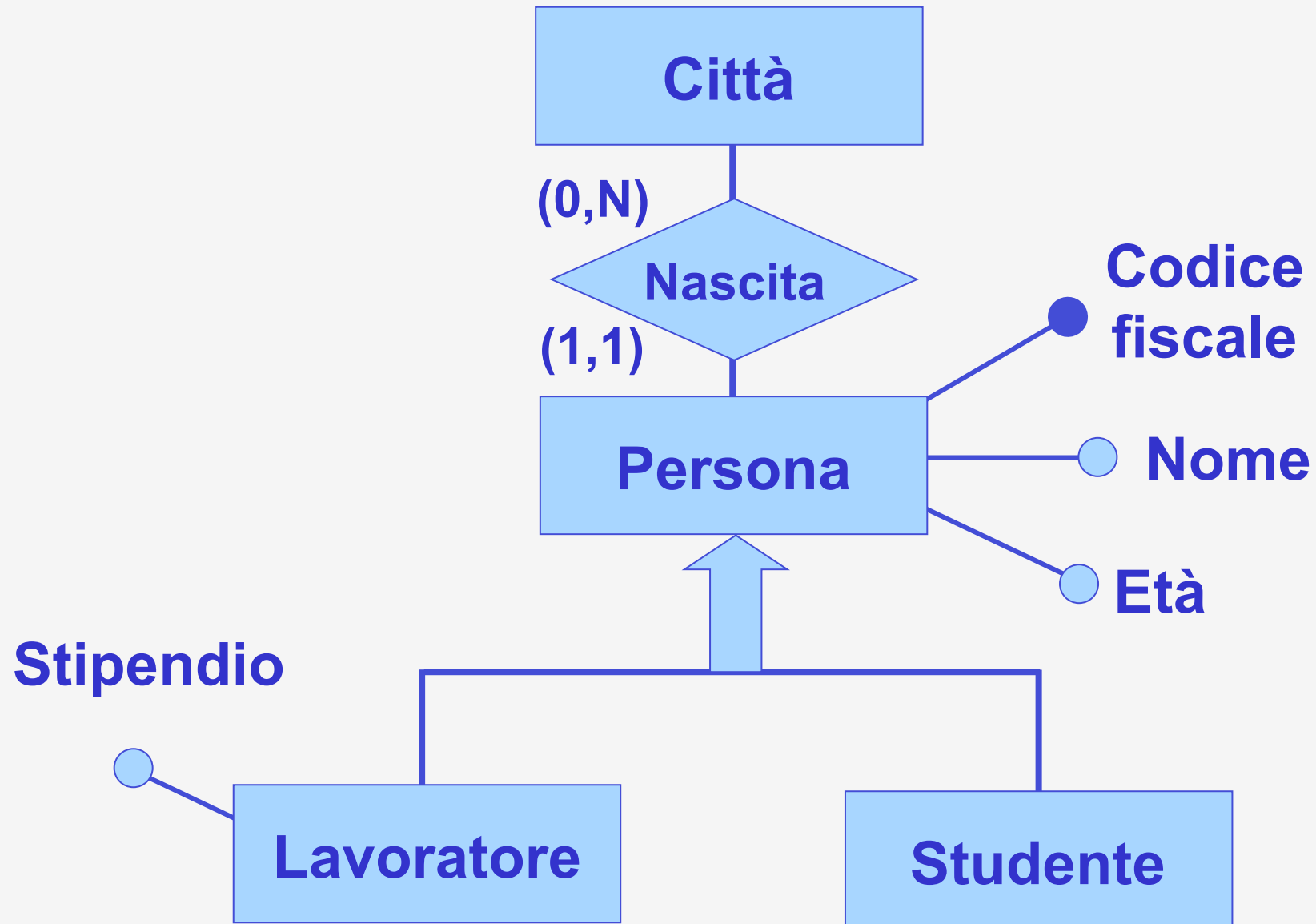


## Proprietà delle generalizzazioni

Se  $E$  (genitore) è generalizzazione di  $E_1, E_2, \dots, E_n$  (figlie):

- ogni proprietà di  $E$  è significativa per  $E_1, E_2, \dots, E_n$
- ogni occorrenza di  $E_1, E_2, \dots, E_n$  è occorrenza anche di  $E$

# Proprietà delle generalizzazioni: esempio



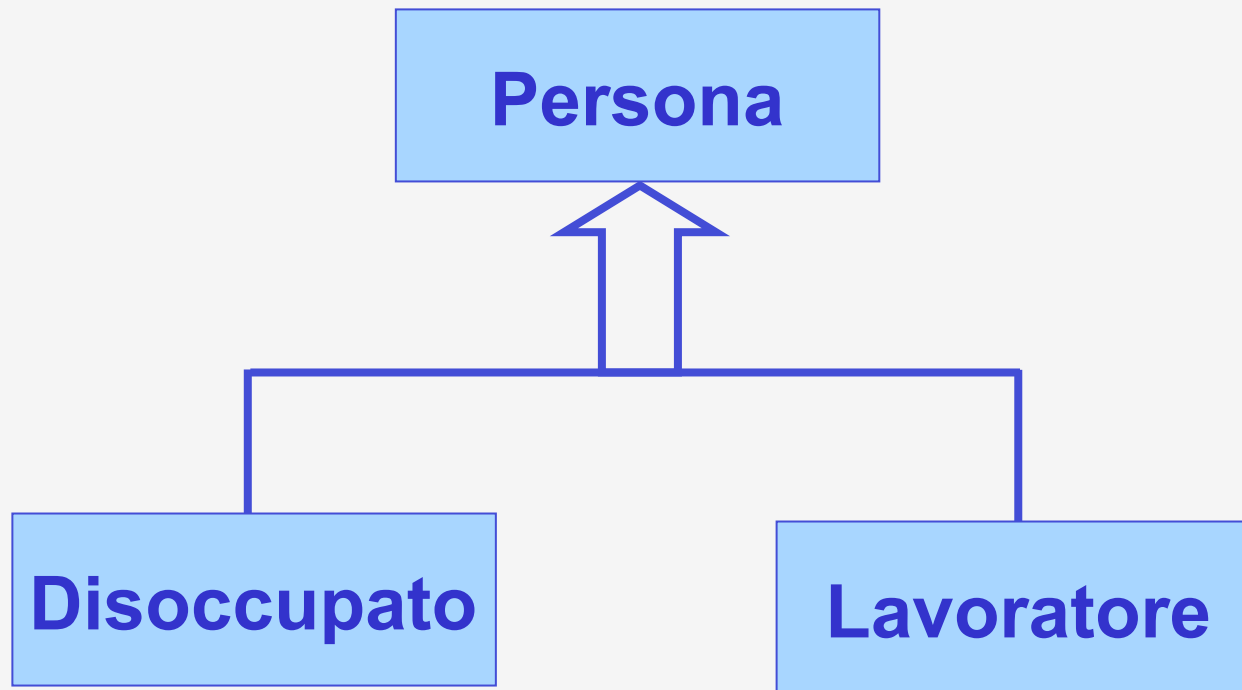
# Ereditarietà

- tutte le proprietà (attributi, relationship, altre generalizzazioni) dell' entità genitore vengono **ereditate** dalle entità figlie e non rappresentate esplicitamente

## Tipi di generalizzazioni

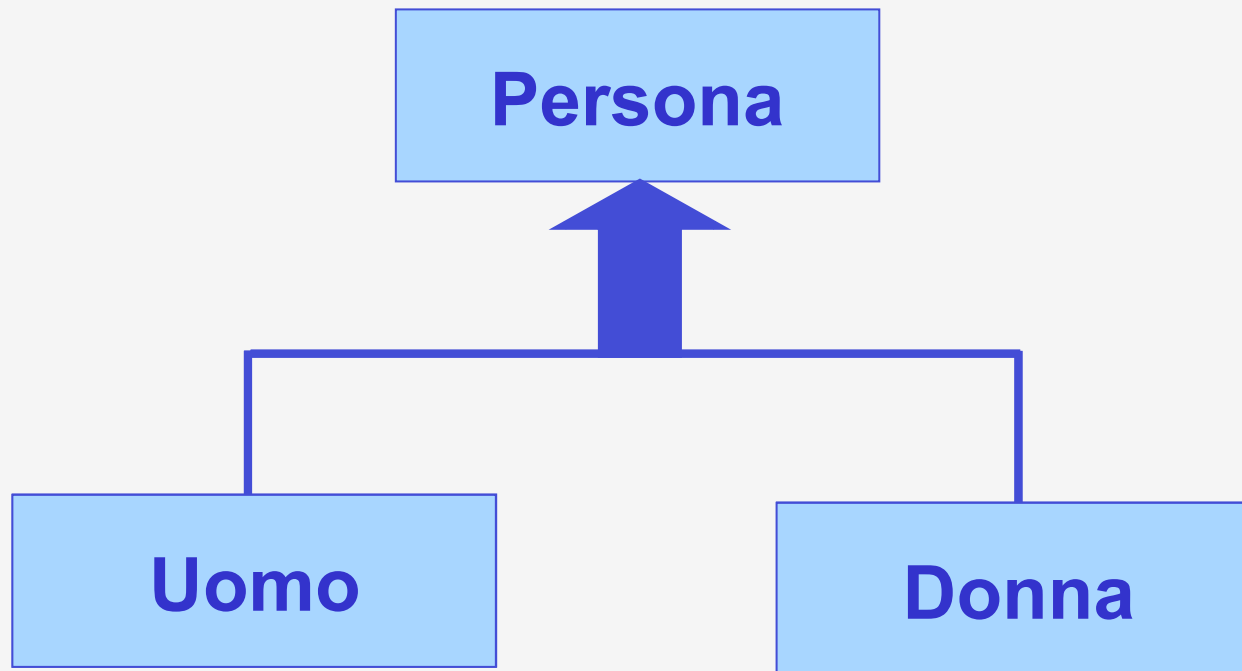
- **totale** se ogni occorrenza dell'entità genitore è occorrenza di almeno una delle entità figlie, altrimenti è **parziale**
- **esclusiva** se ogni occorrenza dell'entità genitore è occorrenza di al più una delle entità figlie, altrimenti è **sovrapposta**
- consideriamo (senza perdita di generalità) solo generalizzazioni esclusive e distinguiamo fra totali e parziali

# Generalizzazione esclusiva parziale





# Generalizzazione esclusiva totale



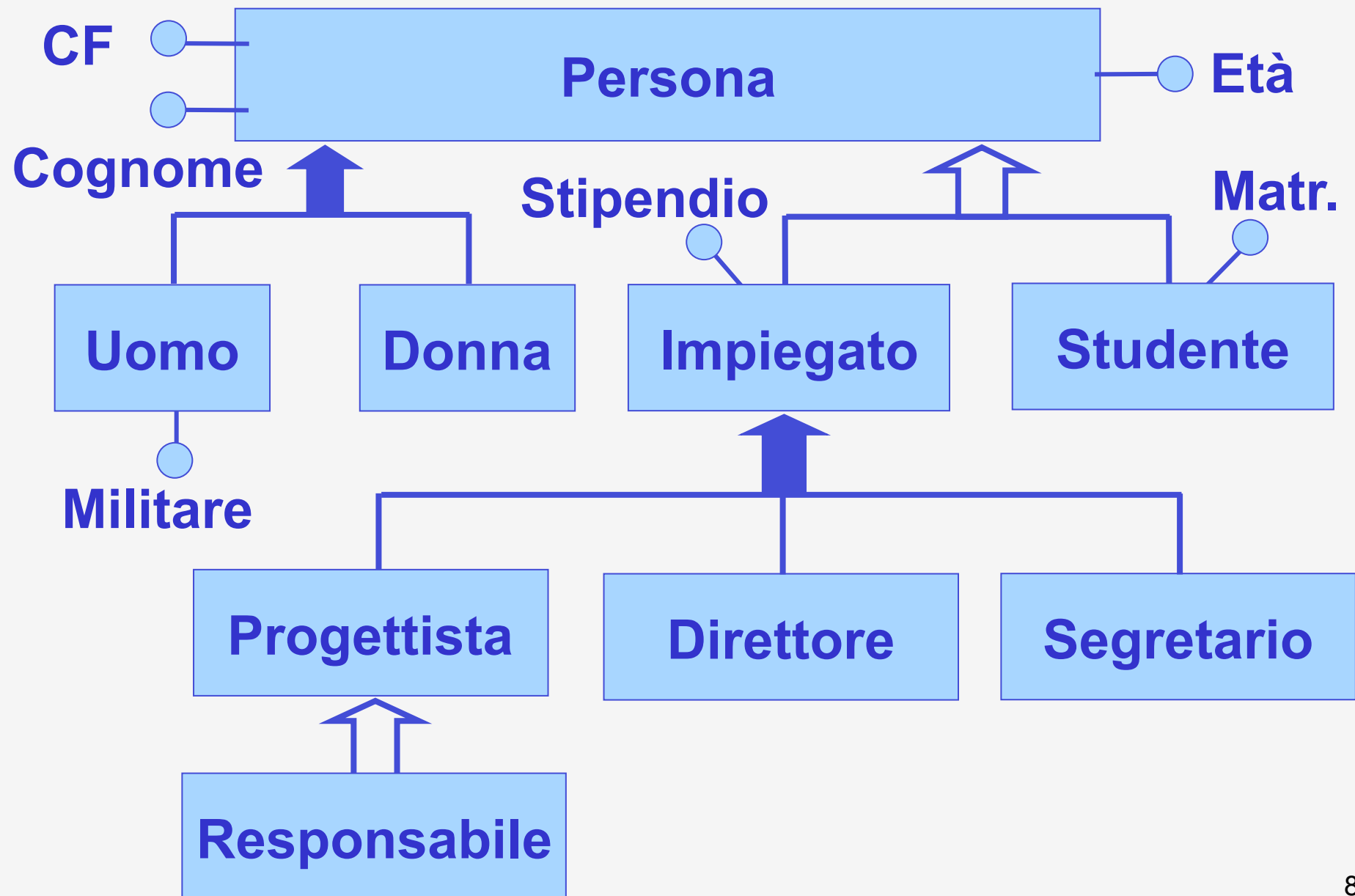
## Altre proprietà

- possono esistere gerarchie a più livelli e multiple generalizzazioni allo stesso livello
- un'entità può essere inclusa in più gerarchie, come genitore e/o come figlia
- se una generalizzazione ha solo un' entità figlia si parla di **sottoinsieme**
- alcune configurazioni non hanno senso
- il genitore di una generalizzazione totale può non avere identificatore, purché ...

## Esercizio

- Le persone hanno CF, cognome ed età; gli uomini anche la posizione militare; gli impiegati hanno lo stipendio e possono essere segretari, direttori o progettisti (un progettista può essere anche responsabile di progetto); gli studenti (che non possono essere impiegati) un numero di matricola; esistono persone che non sono né impiegati né studenti (ma i dettagli non ci interessano)

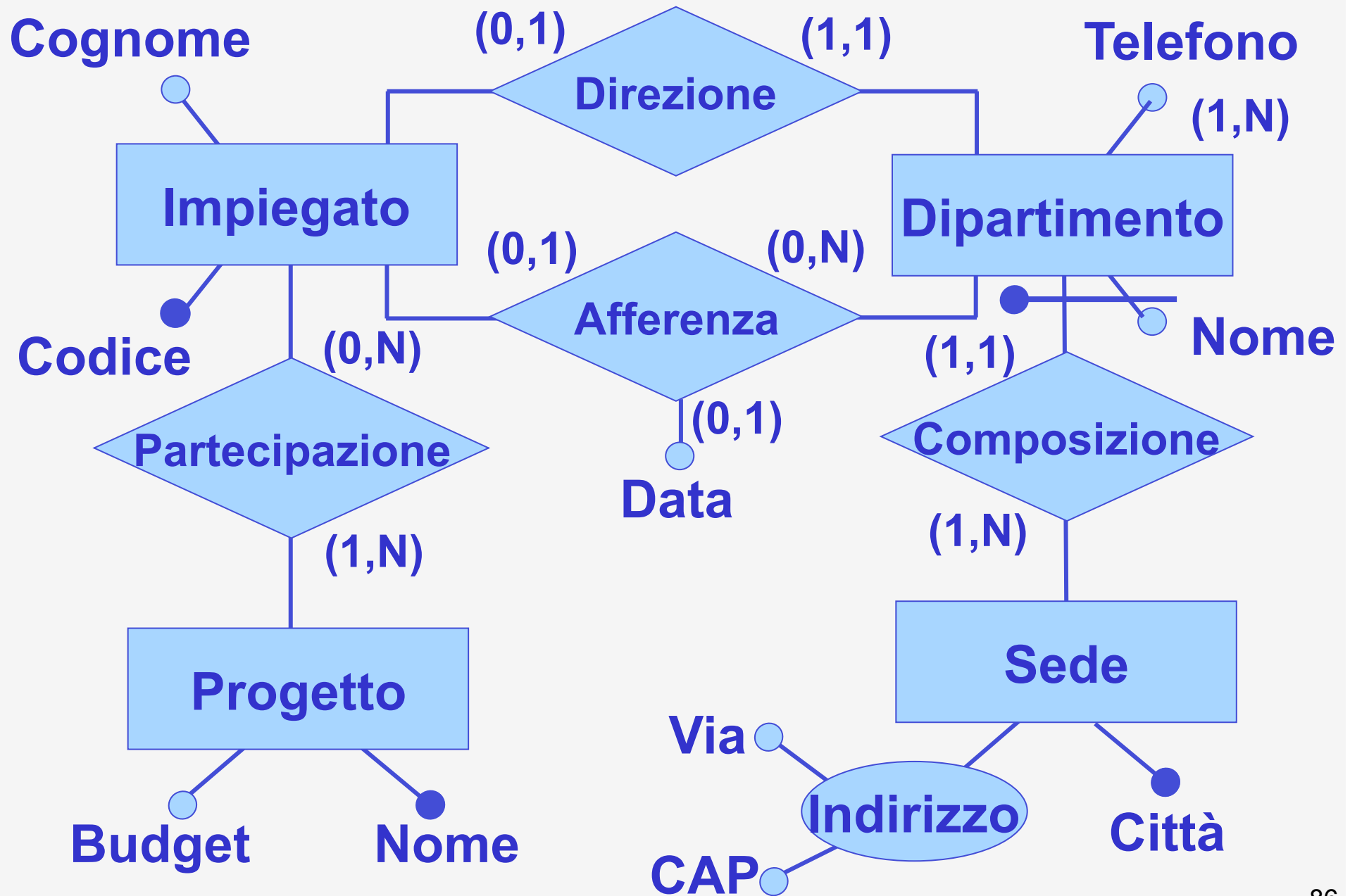
# Schema concettuale



# Documentazione associata agli schemi concettuali

- dizionario dei dati
  - entità
  - relationship
- vincoli non esprimibili

## Schema concettuale (2)



## Dizionario dei dati (entità)

Entità	Descrizione	Attributi	Identificatore
Impiegato	Dipendente dell'azienda	Codice, Cognome, Stipendio	Codice
Progetto	Progetti aziendali	Nome, Budget	Nome
Dipartimento	Struttura aziendale	Nome, Telefono	Nome, Sede
Sede	Sede dell'azienda	Città, Indirizzo	Città

## Dizionario dei dati (relationship)

Relazioni	Descrizione	Componenti	Attributi
Direzione	Direzione di un dipartimento	Impiegato, Dipartimento	
Afferenza	Afferenza a un dipartimento	Impiegato, Dipartimento	Data
Partecipazione	Partecipazione a un progetto	Impiegato, Progetto	
Composizione	Composizione dell'azienda	Dipartimento, Sede	



# Vincoli non esprimibili

## **Vincoli di integrità sui dati**

- |  |
|--|
| (1) Il direttore di un dipartimento deve afferire a tale dipartimento                                      |
| (2) Un impiegato non deve avere uno stipendio maggiore del direttore del dipartimento al quale afferisce   |
| (3) Un dipartimento con sede a Roma deve essere diretto da un impiegato con più di dieci anni di anzianità |
| (4) Un impiegato che non afferisce a nessun dipartimento non deve partecipare a nessun un progetto         |

# Modellazione dei dati in UML

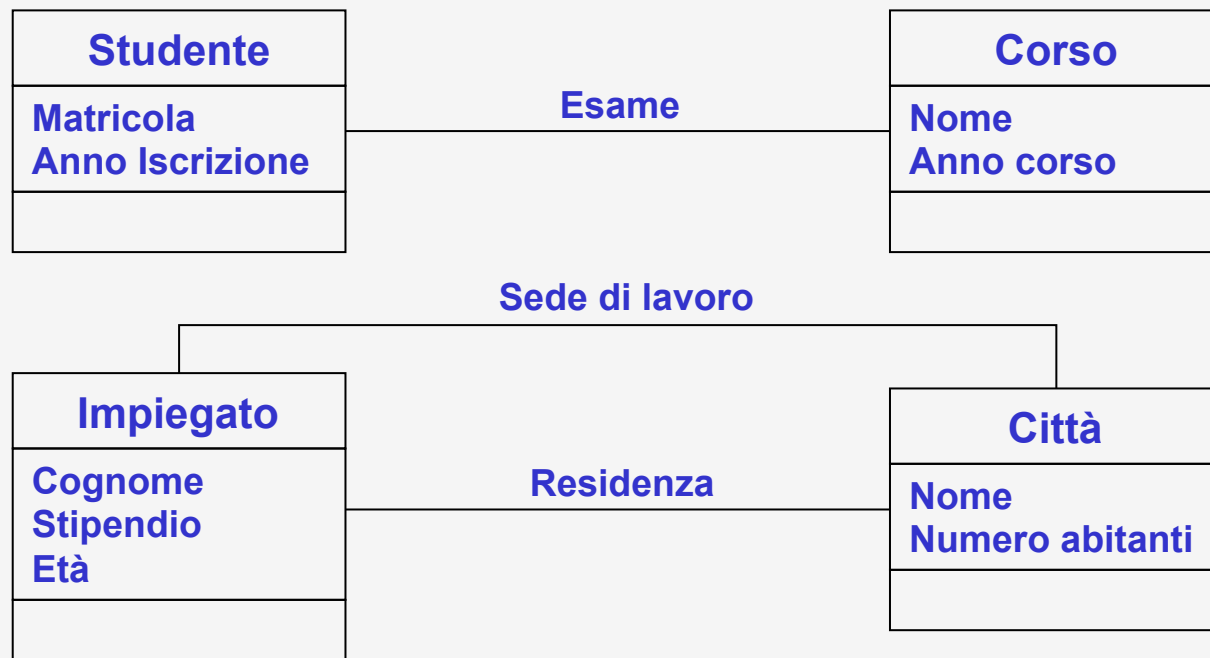
- UML viene talvolta utilizzato in alternativa al modello ER per la rappresentazione concettuale dei dati
- Si fa uso dei diagrammi delle classi
- Cambia la rappresentazione diagrammatica ma non l'approccio alla progettazione
- Vediamo come sia possibile rappresentare schemi concettuali con UML

# Classi

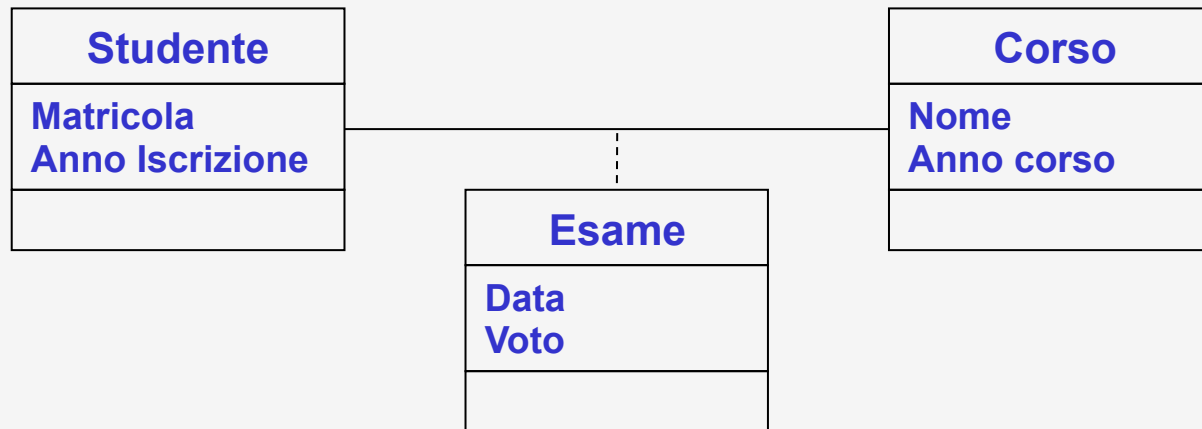
Impiegato
Codice Cognome Stipendio Età

Progetto
Nome Budget Data consegna

# Associazioni

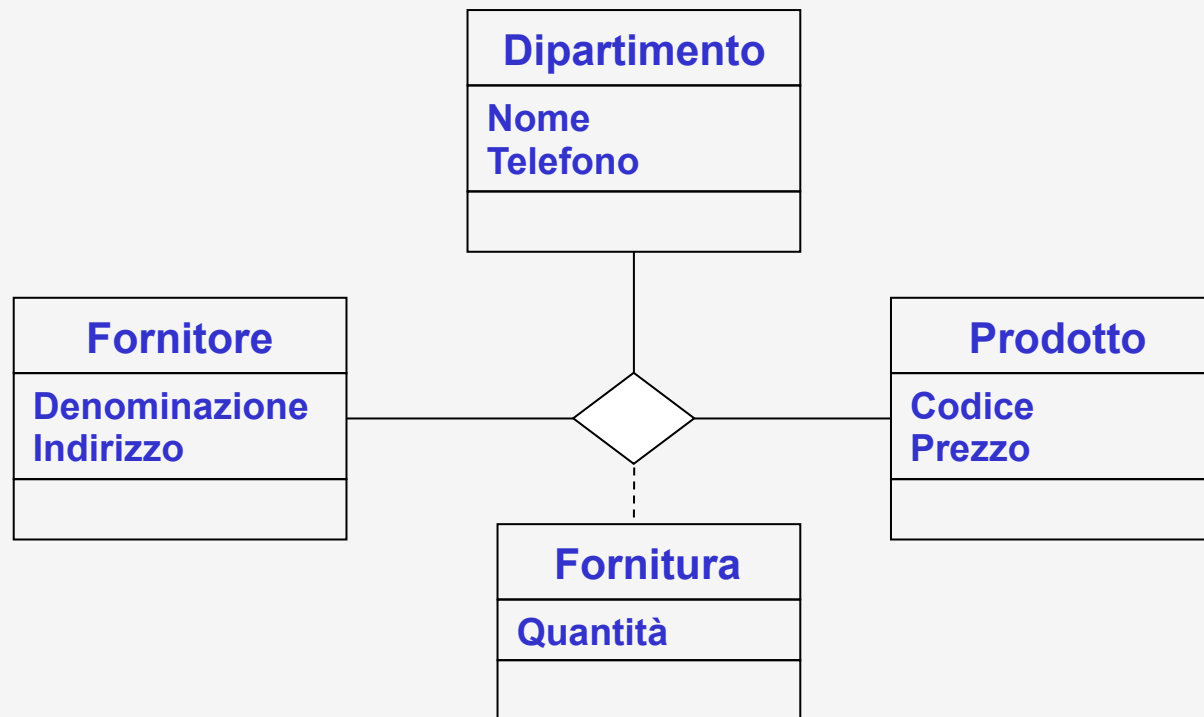


# Classe di associazione



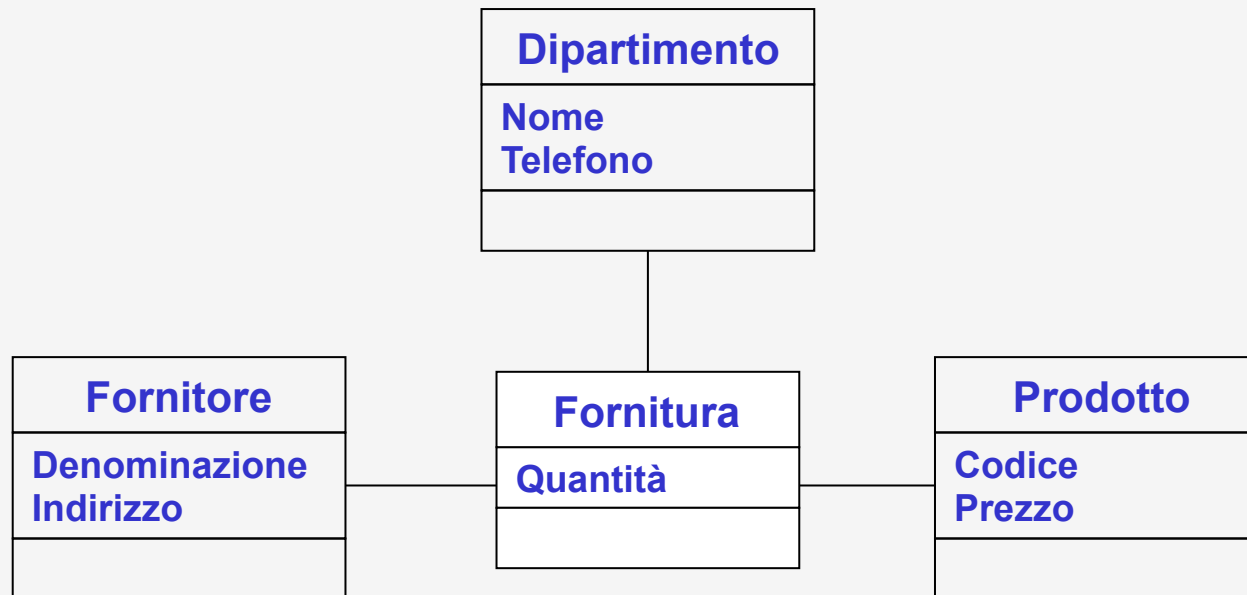
La classe **Esame** è una classe di associazione che usiamo per rappresentare gli attributi **Voto** e **Data** dell'associazione tra la classe **Studente** e la classe **Corso**

# Associazione ternaria



E' rappresentata da un rombo con le classi che vi partecipano (**Fornitore**, **Dipartimento** e **Prodotto**), facendo uso di una classe di associazione (**Fornitura**) per assegnare attributi all'associazione tra queste classi

# Reificazione di associazione



Le associazioni n-arie sono poco usate nei diagrammi delle classi e, per questo, le reifichiamo ovvero trasformiamo l'associazione (**Fornitura**) in una classe legata alle classi originarie con associazioni binarie

# Aggregazione e composizione



- Un **Tecnico** fa parte di un **Team**. Esso può essere rappresentato indipendentemente dal **Team** di cui fa parte (**aggregazione semplice**)



- Una **Filiale** è parte di un'**Azienda**. Essa non può essere rappresentata indipendentemente dall'**Azienda** (**composizione**)



# Associazioni con molteplicità



- Un **Ordine** può avere 0 o al massimo 1 **Fattura** di vendita associata. Viceversa una **Fattura** ha 1 solo **Ordine** di vendita



- L'assenza di molteplicità sottintende 1..1, quindi una **Persona** può essere residente in una ed una sola **Città**. Viceversa una **Città** può avere 0 o al massimo N persone residenti



- Un **Turista** può prenotare 1 o al massimo N viaggi. Viceversa un **Viaggio** può essere prenotato da 0 o al massimo N turisti

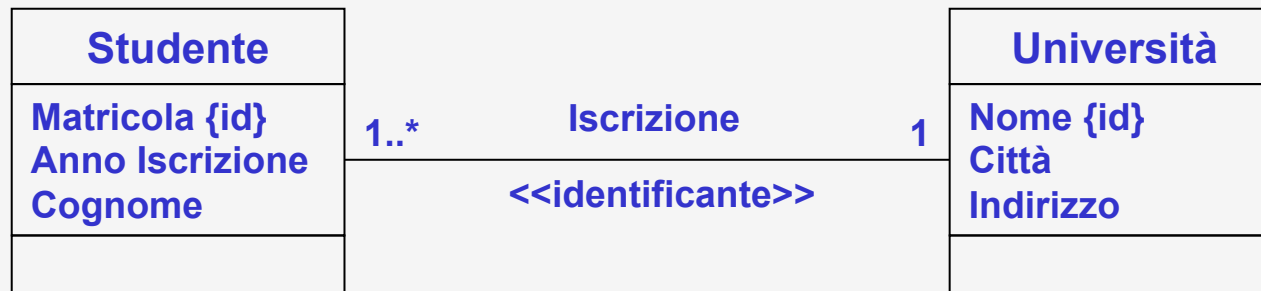
# Identificatori

Automobile
Targa {id} Modello Colore

Persona
Data Nascita {id} Cognome {id} Nome {id} Indirizzo

- **Targa** è identificatore della classe **Automobile**
- L'insieme di attributi **Data Nascita**, **Cognome** e **Nome** costituiscono l'identificatore per la classe **Persona**

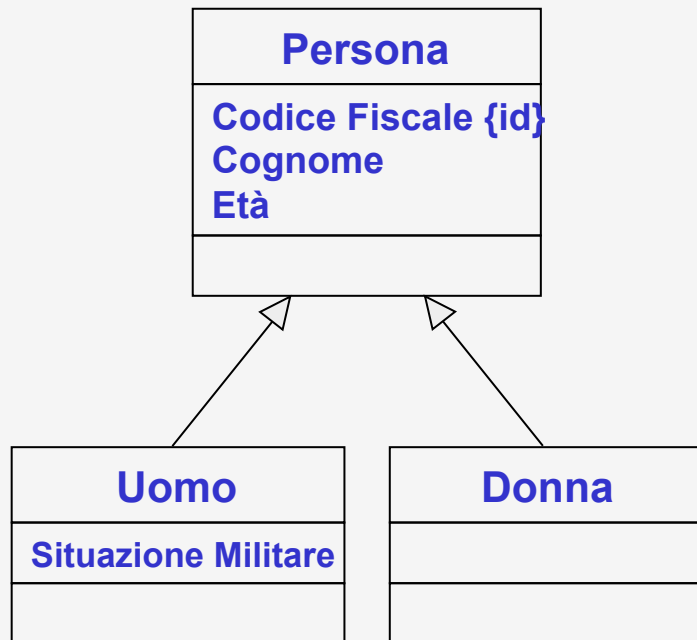
# Identificatore esterno



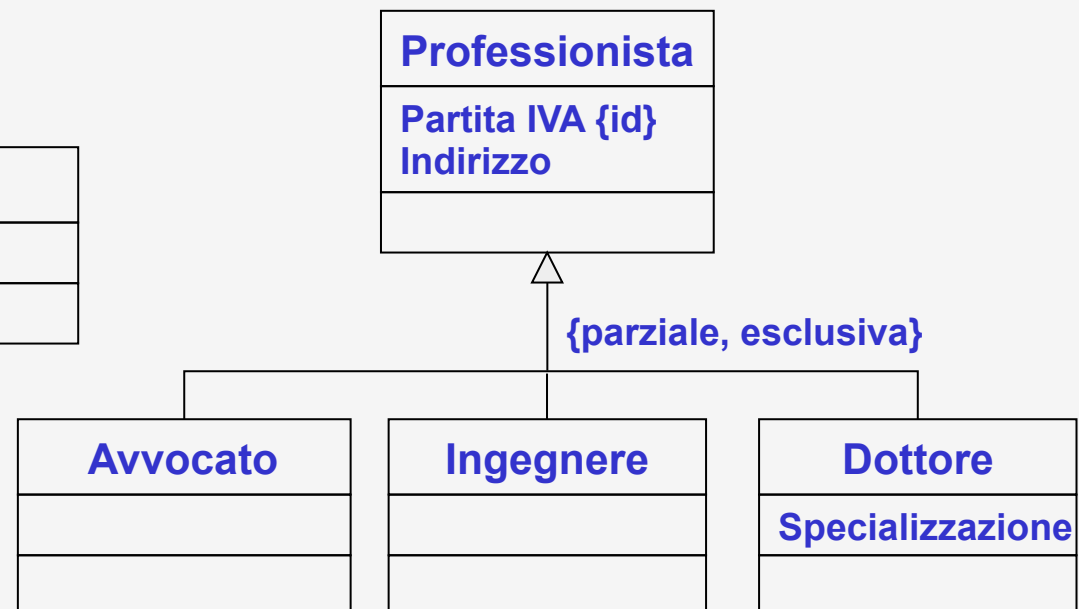
- Lo stereotipo **<<identificante>>** serve ad indicare che l'associazione tra **Studente** e **Università** è appunto identificante in quanto insieme all'attributo **Matricola** identifica uno studente

# Generalizzazioni

## Esclusiva Totale



## Esclusiva Parziale



# Uno schema concettuale in UML

