

# Progettazione logica delle basi di dati

## Ristrutturazione dello schema Entità–Relazione

Prof. Fedeli Massimo - Tutti i diritti riservati

# Perché non basta tradurre lo schema ER

Lo schema Entità Relazione consente di rappresentare costrutti che il modello relazionale non supporta direttamente.

Tali costrutti sono:

- attributi multivalue;
- attributi composti;
- gerarchie ISA;
- generalizzazioni.

Per questo è necessaria una fase di ristrutturazione.

# Il concetto di carico applicativo

Quando si progetta un database è fondamentale sapere come verrà usato.

Il **carico applicativo** descrive:

- quali operazioni vengono eseguite;
- con quale frequenza;
- su quali dati.

Queste informazioni influenzano le scelte progettuali.

# Perché il carico applicativo è importante

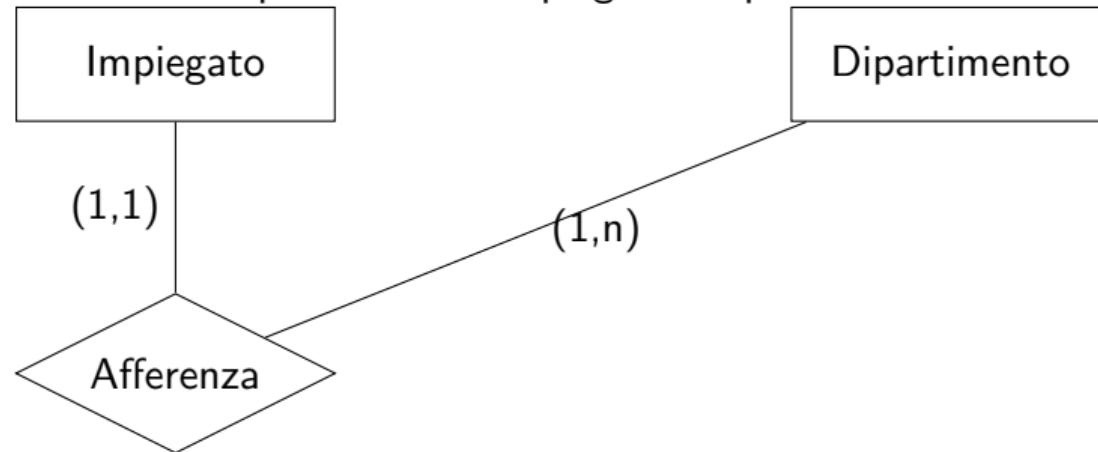
Due schemi equivalenti dal punto di vista dei dati possono avere **prestazioni** molto diverse.

Conoscere il carico applicativo permette di:

- valutare i costi delle operazioni;
- decidere se mantenere o eliminare ridondanze;
- migliorare l'efficienza complessiva del sistema.

# Schema ER di esempio

Consideriamo uno schema semplificato con impiegati e dipartimenti.



Ogni impiegato lavora in un solo dipartimento, mentre un dipartimento può avere più impiegati.

# Ristrutturazione dello schema ER

La ristrutturazione consiste nel modificare lo schema ER per renderlo **compatibile** con il modello relazionale.

Non si perdono informazioni, ma:

- si semplifica la struttura;
- si rendono esplicativi i vincoli;
- si prepara lo schema alla traduzione.

Le principali attività della ristrutturazione sono:

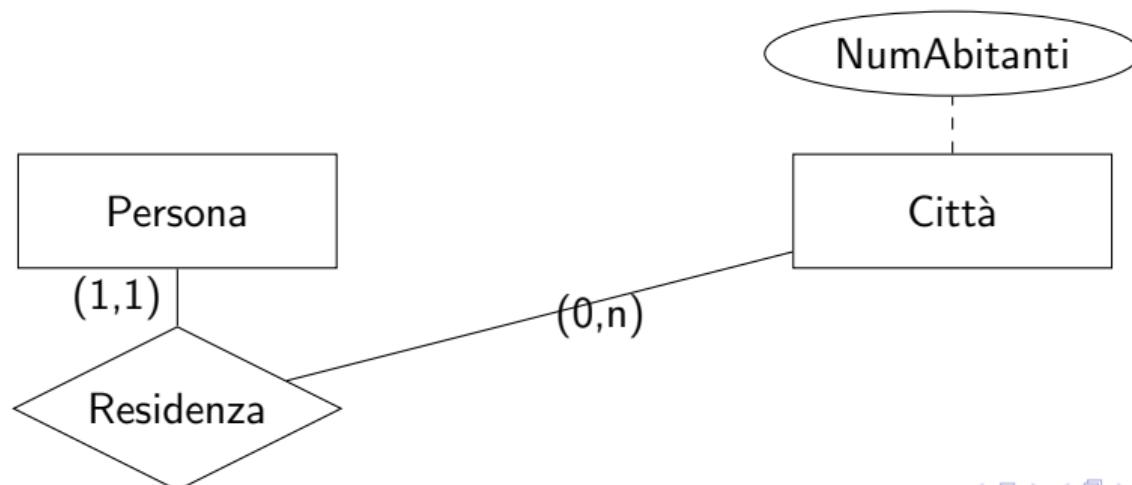
- ① analisi delle ridondanze;
- ② eliminazione degli attributi multivалore;
- ③ eliminazione degli attributi composti;
- ④ eliminazione di ISA e generalizzazioni;
- ⑤ scelta degli identificatori principali.

# Ridondanza: definizione

Una **ridondanza** è un'informazione che può essere derivata da altre informazioni già presenti nello schema.

Esempio:

- **numero di abitanti** di una città;
- **derivabile** dal numero di persone residenti.



**Osservazione:** l'attributo *NumAbitanti* è ridondante perché può essere **calcolato** contando le persone collegate alla città tramite la relazione *Residenza*

# Vantaggi e svantaggi delle ridondanze

Mantenere una ridondanza può avere effetti diversi sulle prestazioni e sulla gestione dei dati.

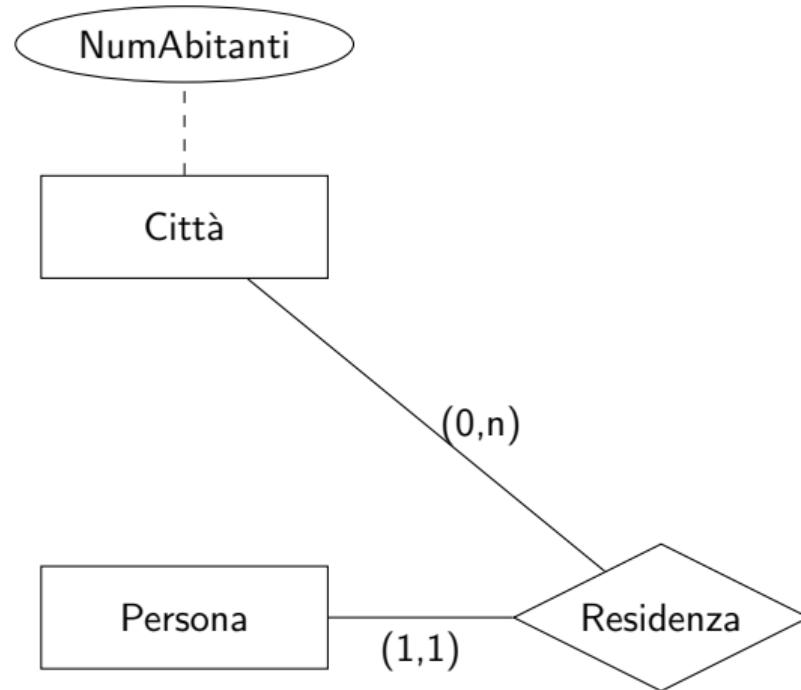
## Vantaggi (lettura):

- interrogazioni più veloci;
- meno accessi ai dati.

## Svantaggi (aggiornamento):

- aggiornamenti più complessi;
- rischio di incoerenza;
- maggiore occupazione di spazio.

**In sintesi:** la ridondanza favorisce le interrogazioni, ma rende più delicata la gestione degli aggiornamenti.



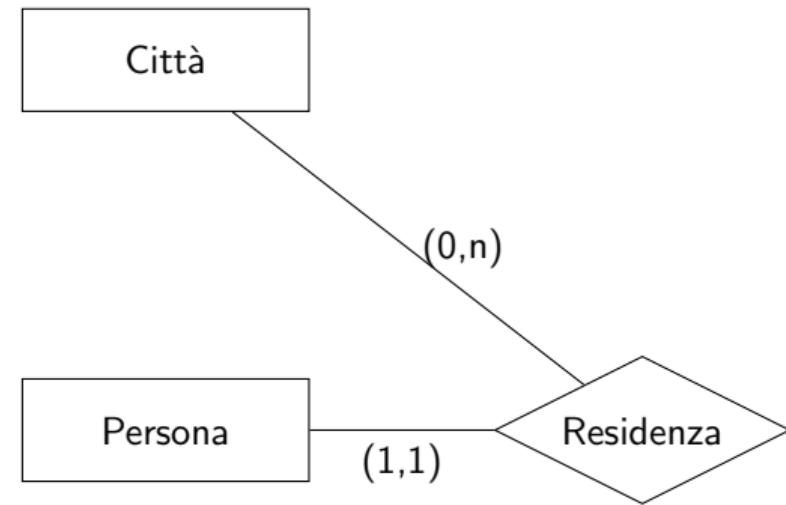
Lettura:

## Esempio di ristrutturazione: schema iniziale

Consideriamo un sistema che gestisce persone e città di residenza.

Nello schema iniziale **non è presente alcuna ridondanza**: il numero di abitanti di una città può essere **calcolato** contando le persone residenti.

- Ogni persona risiede in una sola città
- Una città può avere molte persone
- Il numero di abitanti non è memorizzato



**Osservazione:** lo schema è corretto ma può risultare costoso per interrogazioni frequenti sul numero di abitanti.

# Valutazione dei costi: schema senza ridondanza

Per valutare l'efficienza dello schema analizziamo **volumi dei dati e accessi alle operazioni.**

## Tabella dei volumi

Costrutto	Numero di istanze
Persona	1.000.000
Città	200
Residenza	1.000.000

## Operazione di interesse

*Stampare i dati di una città con il numero di abitanti (eseguita 2 volte al giorno).*

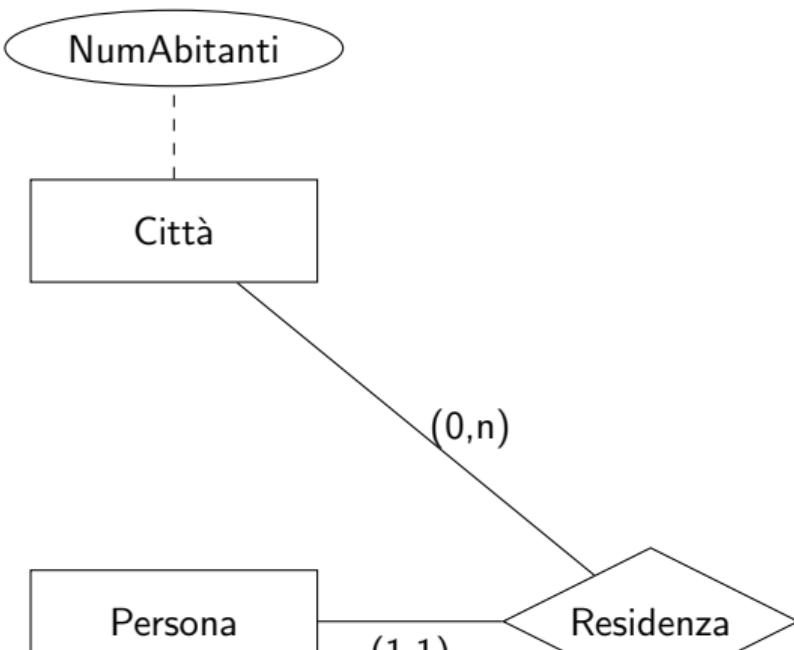
## Tabella degli accessi

Costrutto	Accessi in lettura
Residenza	5.000
Persona	5.000

# Schema ristrutturato con ridondanza

Per migliorare le prestazioni delle interrogazioni, si decide di **introdurre una ridondanza**.

Si aggiunge all'entità *Città* l'attributo *NumAbitanti*, aggiornato ogni volta che una persona cambia residenza.



## Nuova valutazione dei costi

- Lettura numero abitanti:
  - 1 solo accesso alla tabella *Città*
- Inserimento di una persona:
  - 1 scrittura su *Persona*
  - 1 aggiornamento su *Città*

**Conclusione:** meno costi in lettura, più costi in aggiornamento.

# Confronto dei costi: con e senza ridondanza

Confrontiamo ora i costi delle operazioni principali nei due schemi.

## Operazioni considerate

- O1: inserimento di una nuova persona
- O2: lettura dei dati di una città con numero di abitanti

## Confronto degli accessi

Operazione	Senza ridondanza	Con ridondanza
O1 – Inserimento persona	1 scrittura	2 scritture
O2 – Lettura abitanti	10.000 letture	1 lettura

## Interpretazione

- la ridondanza aumenta il costo degli aggiornamenti;
- riduce drasticamente il costo delle interrogazioni;
- conviene se le letture sono molto più frequenti delle scritture.

**Conclusione progettuale:** la scelta dipende dal carico applicativo, non solo dalla struttura dello schema.

## Contro-esempio: carico dominato dagli aggiornamenti

Consideriamo ora un contesto applicativo diverso, in cui le **operazioni di aggiornamento sono molto frequenti**, mentre le interrogazioni aggregate sono rare.

Scenario tipico:

- sistema anagrafico o gestionale;
- frequenti inserimenti, cancellazioni e cambi di residenza;
- raramente si richiede il numero totale di abitanti.

In questo caso la presenza di una ridondanza può diventare **uno svantaggio**.

# Valutazione dei costi: aggiornamenti frequenti

Analizziamo volumi e operazioni nel nuovo scenario.

## Tabella dei volumi

Costrutto	Numero di istanze
Persona	500.000
Città	200
Residenza	500.000

## Operazioni di interesse

- O1: inserimento o modifica residenza (3.000 operazioni al giorno)
- O2: lettura numero abitanti di una città (1 volta al giorno)

Il carico applicativo è chiaramente sbilanciato sugli aggiornamenti.

# Confronto dei costi: eliminare la ridondanza

Confrontiamo i costi nei due schemi.

Operazione	Con ridondanza	Senza ridondanza
O1 – Aggiornamento residenza	2 scritture	1 scrittura
O2 – Lettura abitanti	1 lettura	5.000 letture

## Valutazione complessiva

- gli aggiornamenti sono molto frequenti;
- la lettura aggregata è rara;
- il costo aggiuntivo degli aggiornamenti non è giustificato.

**Decisione progettuale:** in questo scenario è preferibile **eliminare il campo calcolato** e calcolare il valore solo quando necessario.

# Attributi multivaleore

Un **attributo multivaleore** è un attributo che può assumere più valori per una stessa entità.

Esempi tipici:

- numeri di telefono;
- indirizzi email;
- lingue conosciute.

# Perché gli attributi multivalore non sono ammessi

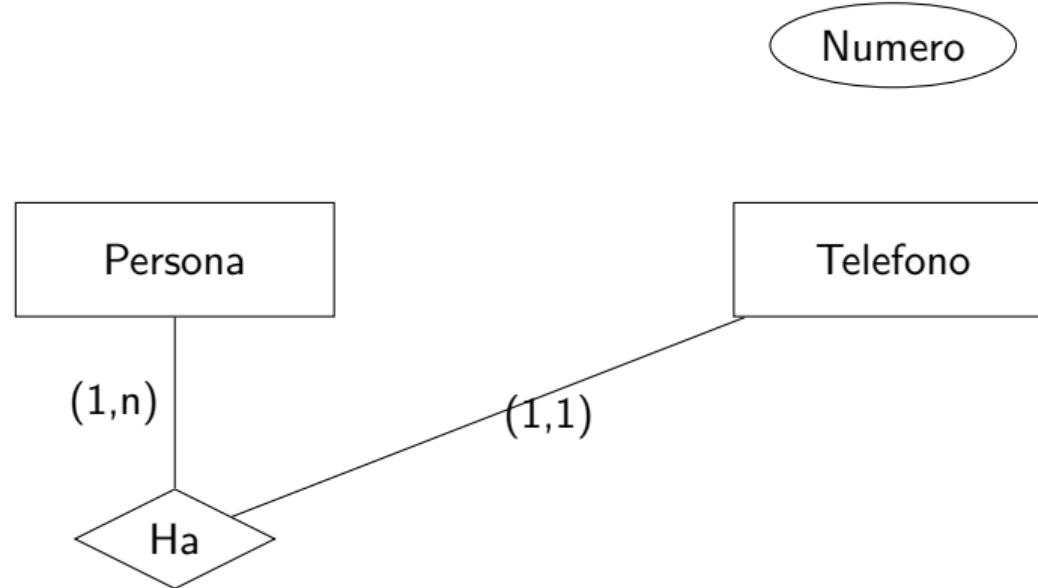
Nel modello relazionale:

- ogni campo di una tabella deve contenere un solo valore;
- non sono ammesse liste o insiemi di valori.

Per questo gli attributi multivalore devono essere eliminati.

## Eliminazione di un attributo multivalue

L'attributo multivalore viene trasformato in una relazione separata.



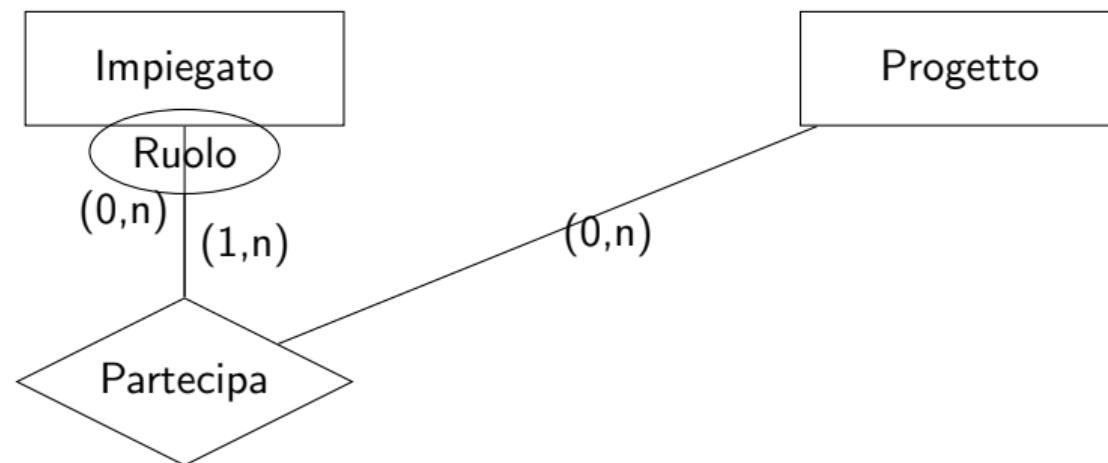
Ogni valore dell'attributo diventa una nuova istanza.

# Attributi multivалore di una relazione

Un attributo multivалore può appartenere anche a una **relazione**, non solo a un'entità.

Esempio:

- un impiegato partecipa a un progetto;
- per ogni partecipazione possono esserci più *ruoli* svolti.

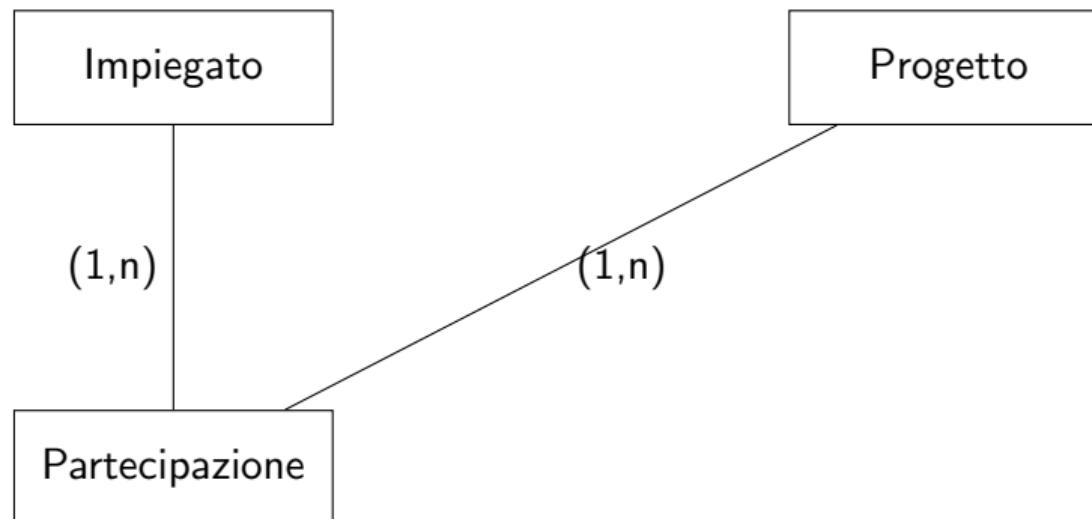


**Problema:** il modello relazionale non ammette attributi multivалore, nemmeno quando sono associati a una relazione

## Eliminazione: trasformare la relazione in entità

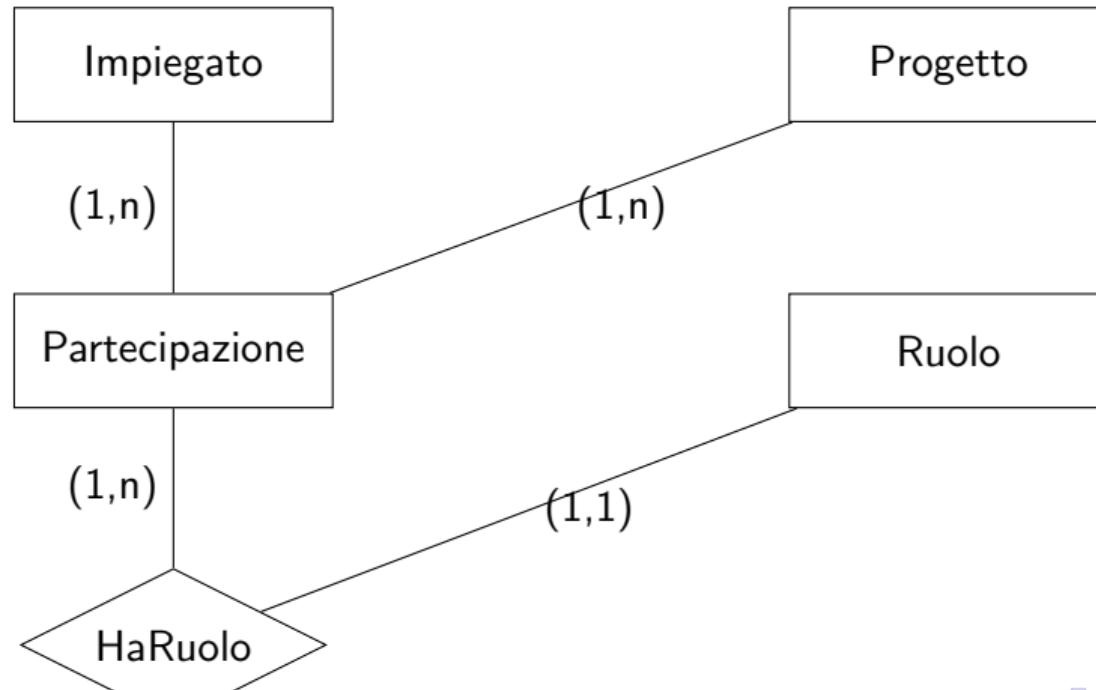
Per eliminare un attributo multivalore di una relazione, è necessario **trasformare la relazione stessa in un'entità**.

La relazione *Partecipa* diventa una nuova entità autonoma, che rappresenta ogni singola partecipazione.



# Eliminazione completa dell'attributo multivalore

A questo punto l'attributo multivalore *Ruolo* può essere eliminato trasformandolo in una relazione separata.



# Eliminazione di attributi multivalore di una relazione

## Procedura riassuntiva in 3 passi

- ① **Individuare l'attributo multivale**re associato alla relazione (es. *Ruolo* nella relazione *Partecipa*).
- ② **Trasformare la relazione in un'entità** ogni istanza della relazione diventa un oggetto autonomo (es. *Partecipa* → *Partecipazione*).
- ③ **Trasformare l'attributo multivale**re in **una relazione** introducendo una nuova entità per il dominio dei valori (es. *Ruolo* come entità collegata a *Partecipazione*).

### Risultato

Lo schema non contiene più attributi multivalore ed è completamente compatibile con il modello relazionale.

# Attributi composti

Un **attributo composto** è formato da più sotto-attributi.

Esempio:

- Indirizzo = Via, Numero, CAP.

Nel modello relazionale gli attributi devono essere atomici.

# Eliminazione degli attributi composti

Un attributo composto viene eliminato:

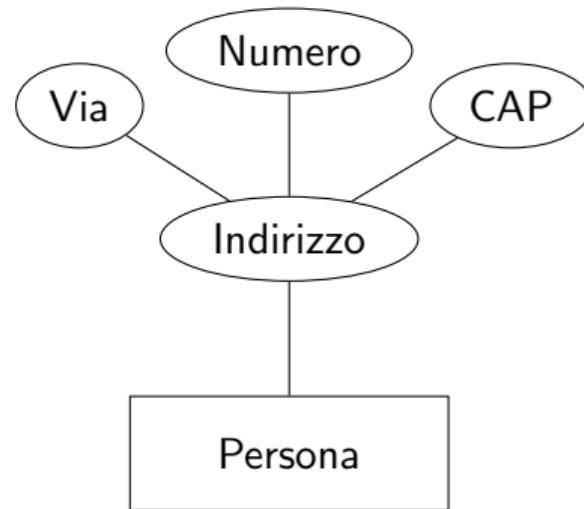
- scomponendolo nei suoi **attributi elementari**;
- oppure trasformandolo in una **nuova entità**.

La scelta dipende dalla cardinalità e dall'uso dell'attributo.

## Esempio 1: scomposizione di un attributo composto

Consideriamo l'attributo composto *Indirizzo* associato all'entità *Persona*.

L'attributo *Indirizzo* è formato da più parti: *Via*, *Numero* e *CAP*.

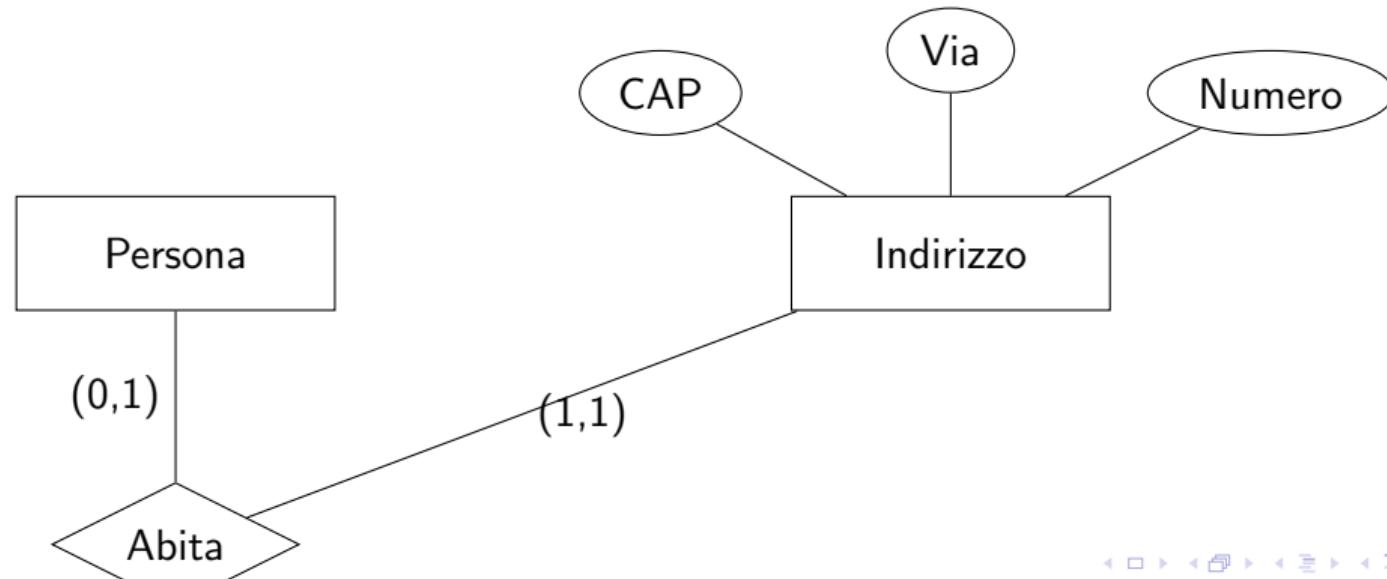


**Trasformazione:** l'attributo composto viene eliminato e i suoi componenti diventano attributi diretti dell'entità.

## Esempio 2: attributo composto trasformato in entità

Se l'attributo composto è opzionale oppure viene riutilizzato, può essere opportuno trasformarlo in una nuova entità.

Esempio: una persona può avere *al più un indirizzo*, ma l'indirizzo è un'informazione autonoma.



# Motivazioni della scelta progettuale

Quando si elimina un attributo composto, la scelta della trasformazione non è arbitraria, ma dipende da precise **motivazioni progettuali**.

## Motivazioni per la scomposizione in attributi elementari

- l'attributo è sempre presente per ogni istanza dell'entità;
- i componenti sono strettamente legati e usati insieme;
- non è necessario gestire l'attributo separatamente;
- si vuole mantenere uno schema semplice e compatto.

## Motivazioni per la trasformazione in nuova entità

- l'attributo è opzionale o può mancare;
- i suoi componenti possono essere usati indipendentemente;
- l'attributo rappresenta un concetto autonomo del dominio;
- è utile evitare valori nulli nell'entità principale.

## Conclusione

Le **relazioni ISA** rappresentano specializzazioni tra entità.

Esempio:

- Studente è una Persona;
- Docente è una Persona.

Nel modello ER sono molto naturali.

# Problema delle ISA nel modello relazionale

Il modello relazionale non supporta direttamente le gerarchie.

È quindi necessario:

- eliminare le ISA;
- esprimere i vincoli in modo esplicito.

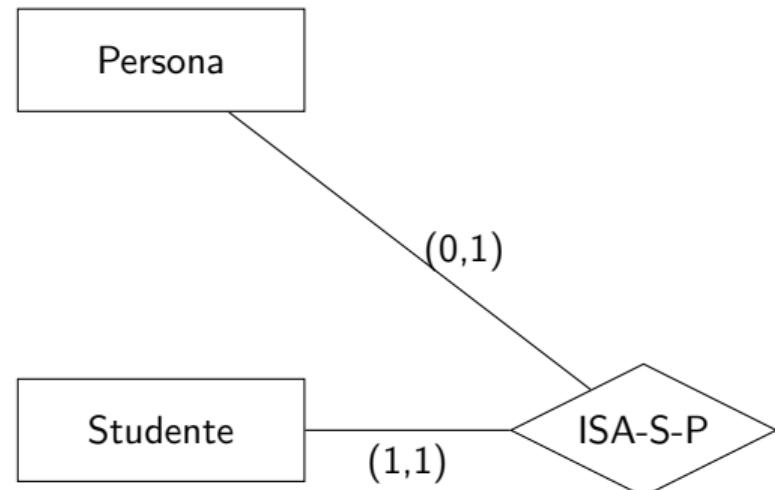
# Eliminazione delle relazioni ISA

La relazione ISA viene sostituita da una **relazione binaria** tra l'entità figlia e l'entità padre.

Dopo la trasformazione:

- le entità diventano **disgiunte**;
- la specializzazione non è più implicita;
- la gerarchia è rappresentata tramite relazioni.

**Osservazione:** i vincoli di specializzazione (vincoli ISA) vengono espressi separatamente.



Studente è specializzazione di Persona

# Perché eliminare la ISA senza collassare le entità

L'eliminazione di una relazione ISA **non** comporta l'unione delle entità coinvolte in un'unica entità.

## Perché non collassare le entità in una sola

- le entità figlie possono avere attributi diversi e specifici;
- non tutte le istanze dell'entità padre appartengono alle entità figlie;
- il collasso introdurrebbe molti valori nulli;
- si perderebbe l'informazione sulla specializzazione.

## Perché usare una relazione binaria

- mantiene separati i concetti del dominio;
- rende esplicita la specializzazione;
- consente di esprimere correttamente i vincoli ISA;
- prepara lo schema alla traduzione nel modello relazionale.

## Idea chiave

# Quando è opportuno collassare le entità in una relazione ISA

In alcuni casi particolari, le entità collegate da una relazione ISA possono essere **collassate in un'unica entità**.

## Condizioni necessarie per il collasso

- tutte le istanze dell'entità padre appartengono a una sola entità figlia (generalizzazione **completa**);
- le entità figlie sono **mutuamente esclusive** (disgiunte);
- gli attributi specifici delle entità figlie sono pochi;
- il numero di valori nulli rimane contenuto.

## Modalità di collasso

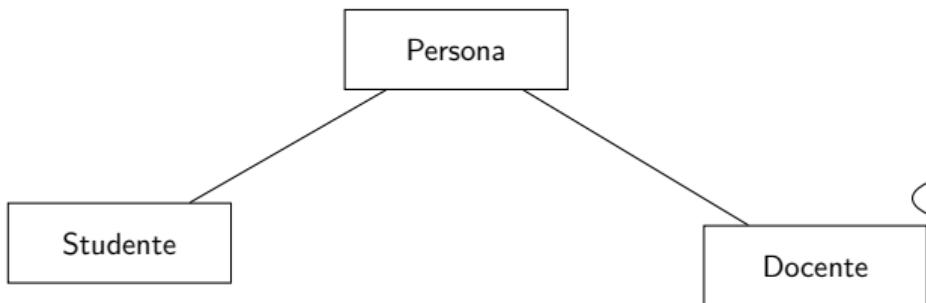
- si crea un'unica entità con tutti gli attributi;
- si introduce un attributo *tipo* (discriminante);
- il valore del discriminante determina quali attributi sono significativi.

## Avvertenza

# Esempio grafico: collasso di una relazione ISA

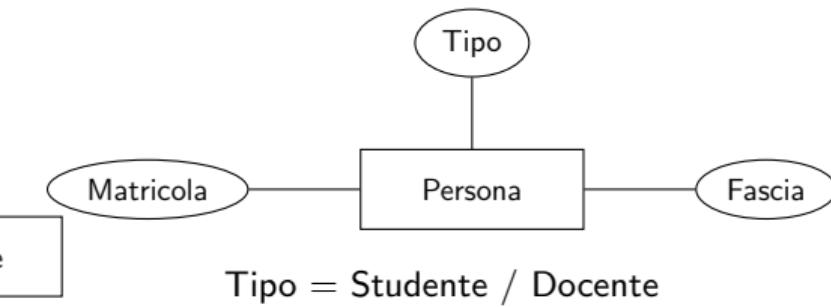
Consideriamo una generalizzazione **completa e disgiunta**, in cui ogni istanza dell'entità padre appartiene esattamente a una sola entità figlia.

**Schema con ISA**



Generalizzazione completa e disgiunta

**Schema dopo il collasso**



$\text{Tipo} = \text{Studente} / \text{Docente}$

**Osservazione:** il collasso è possibile perché ogni persona è *o studente o docente*, senza eccezioni.

# Gestione delle gerarchie ISA: le tre strategie possibili

Quando si incontra una relazione ISA, esistono **tre strategie** di ristrutturazione possibili.

- ① **Collassare le entità figlie nel padre** (un'unica entità con discriminante);
- ② **Sostituire la ISA con relazioni binarie** mantenendo padre e figlie separate;
- ③ **Eliminare l'entità padre** trasferendo i suoi attributi alle entità figlie.

## Obiettivo

Scegliere la strategia che rappresenta meglio il dominio applicativo ed è più adatta all'uso dei dati.

## Guida decisionale: domande da porsi

Per scegliere la strategia corretta, è utile rispondere alle seguenti domande, in ordine.

- ① La generalizzazione è completa?** Tutte le istanze del padre appartengono a una figlia?
- ② Le entità figlie sono disgiunte?** Ogni istanza appartiene a una sola entità figlia?
- ③ Gli attributi specifici delle figlie sono pochi?**
- ④ L'entità padre ha un ruolo autonomo nel dominio?**

Le risposte a queste domande indirizzano la scelta progettuale.

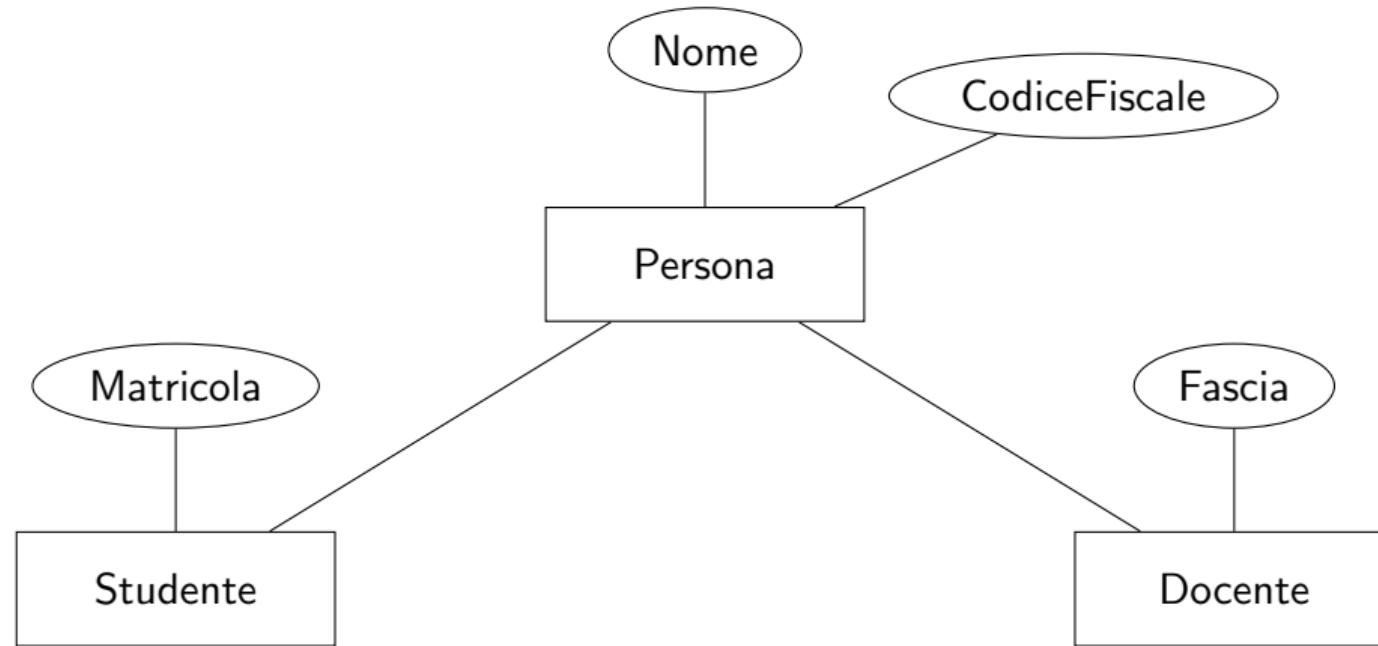
- **Se la generalizzazione è completa e disgiunta**
  - e gli attributi specifici sono pochi → **collassare le entità figlie nel padre**;
- **Se il padre rappresenta un concetto autonomo**
  - oppure la generalizzazione è parziale → **sostituire la ISA con relazioni binarie**;
- **Se l'entità padre non ha significato autonomo**
  - e serve solo a raccogliere attributi comuni → **eliminare il padre e trasferire gli attributi**;

## Regola d'oro

Non scegliere in base alla semplicità dello schema, ma in base al **significato dei dati**.

## Schema iniziale: gerarchia ISA

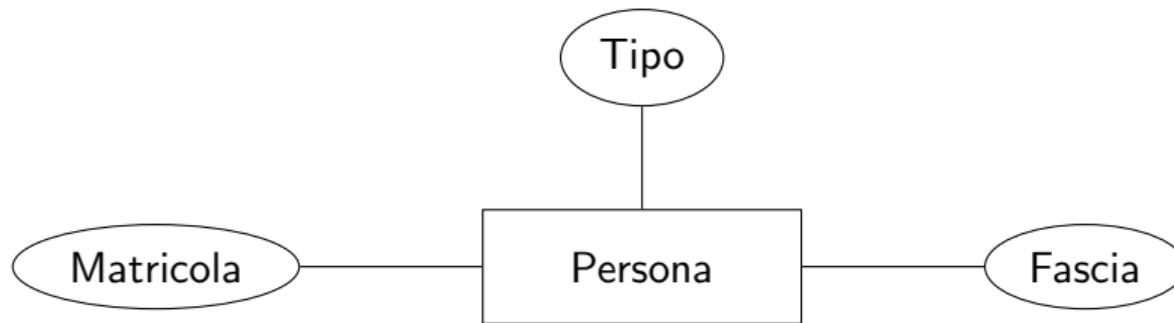
Consideriamo uno schema con una relazione di specializzazione tra un'entità padre e due entità figlie.



Osservazione:

# Esempio 1: collasso delle entità figlie nel padre

**Caso:** generalizzazione completa e disgiunta, con pochi attributi specifici.

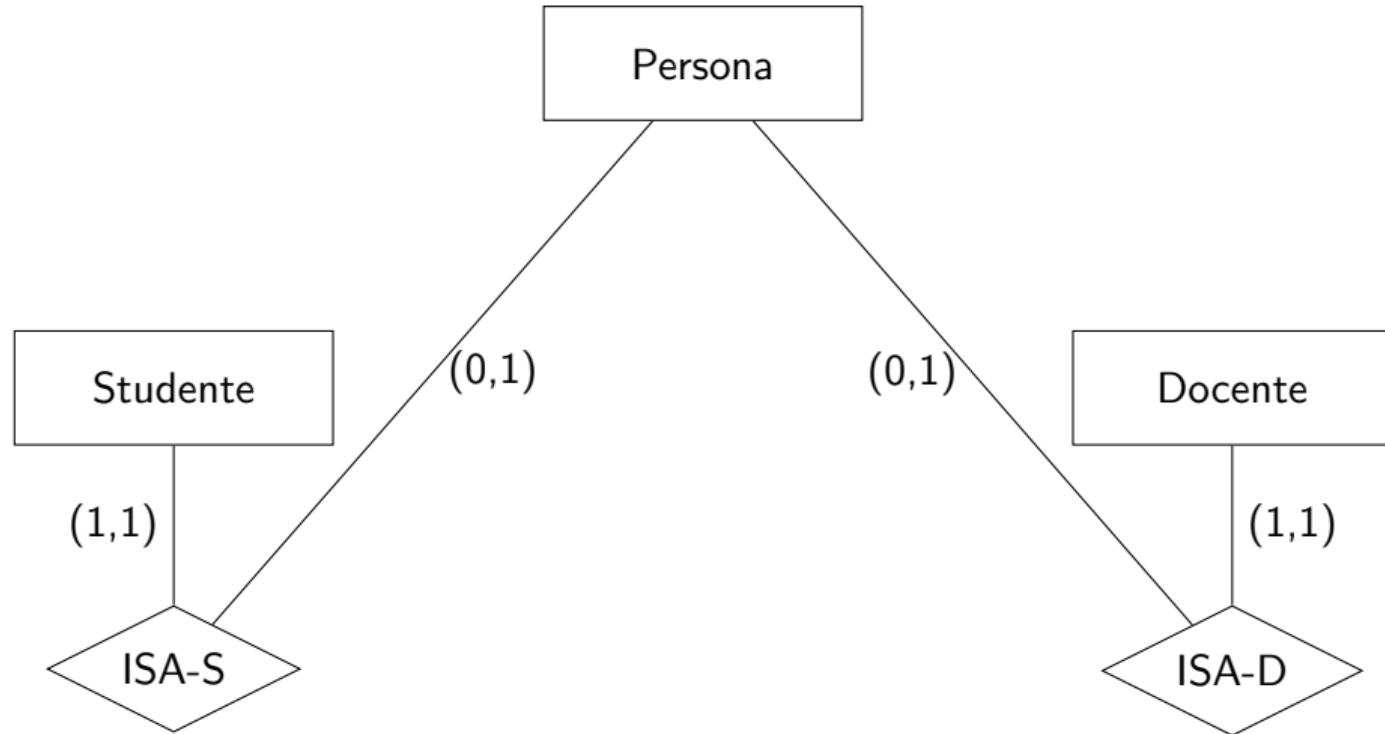


## Spiegazione:

- tutte le persone sono o studenti o docenti;
- un attributo *Tipo* discrimina il ruolo;
- soluzione compatta, ma con possibili valori nulli.

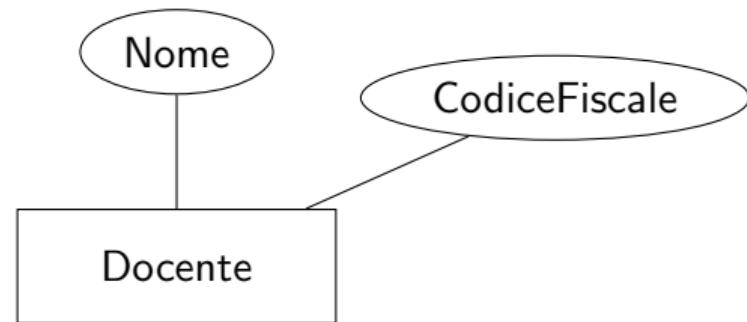
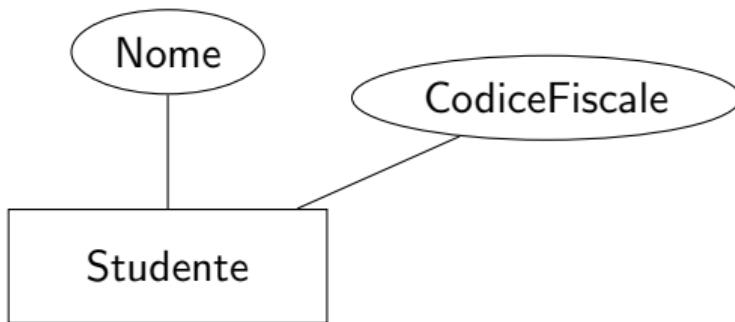
## Esempio 2: sostituzione della ISA con relazioni binarie

**Caso:** entità padre con significato autonomo e generalizzazione non necessariamente completa.



## Esempio 3: eliminazione dell'entità padre

**Caso:** l'entità padre non ha significato autonomo e serve solo a raccogliere attributi comuni.



### Spiegazione:

- gli attributi del padre sono duplicati;
- schema più semplice;
- maggiore ridondanza, minore espressività.

Un **identificatore** è un insieme di attributi che permette di distinguere univocamente ogni istanza di un'entità.

Ogni entità deve avere:

- almeno un identificatore;
- un identificatore principale.

# Scelta dell'identificatore principale

L'identificatore principale dovrebbe essere:

- semplice;
- stabile nel tempo;
- usato frequentemente nelle operazioni.

Se necessario, si introduce un **codice artificiale**.

Al termine della ristrutturazione, lo schema ER:

- non contiene attributi multivalue;
- non contiene attributi composti;
- non contiene ISA o generalizzazioni;
- assegna un identificatore principale a ogni entità.

Lo schema ER ristrutturato è il punto di partenza per la traduzione nel modello relazionale.

Nella fase successiva:

- entità → tabelle;
- relazioni → tabelle;
- attributi → colonne.

La ristrutturazione dello schema ER è una fase fondamentale della progettazione logica.

Permette di:

- mantenere il significato dei dati;
- rispettare le regole del modello relazionale;
- progettare database corretti ed efficienti.

# Riferimenti bibliografici

I contenuti presentati si basano sui principali testi e materiali di riferimento per la progettazione delle basi di dati.

- C. Batini, S. Ceri, S. Navathe, *Fondamenti di basi di dati*, McGraw-Hill Education.
- A. Silberschatz, H. Korth, S. Sudarshan, *Database System Concepts*, McGraw-Hill.
- P. Atzeni, S. Ceri, S. Paraboschi, R. Torlone, *Basi di dati – Modelli e linguaggi di interrogazione*, McGraw-Hill.
- Materiale didattico e dispense di corso sulla progettazione concettuale e logica delle basi di dati.

I diagrammi e gli esempi sono stati adattati a fini didattici.