

# Dal Modello Entità-Relazione allo Schema Logico

## Progettazione di Database

Prof. Fedeli Massimo

ITS 4.0 - Tutti i diritti riservati

A.S. 2025/26

# Contenuti

- 1 Introduzione alla Progettazione Logica
- 2 Modello Logico Preciso
- 3 Ristrutturazione dello Schema E-R
- 4 Eliminazione Gerarchie
- 5 Traduzione al Modello Relazionale
- 6 Traduzione delle Relazioni
- 7 Relazioni Ternarie
- 8 Relazioni Ridondanti

# Progettazione Logica: Obiettivo

## Obiettivo

Costruire uno **schema logico** in grado di descrivere, in maniera corretta ed efficiente, tutte le informazioni contenute nello schema Entità-Relazione prodotto nella fase di progettazione concettuale.

## Importante

Non è una semplice traduzione da un modello a un altro!

# Progettazione Logica: Ristrutturazione

Prima di passare allo schema logico, lo schema E-R va **ristrutturato** per:

- ① **Semplificare** la traduzione
- ② **Ottimizzare** il progetto

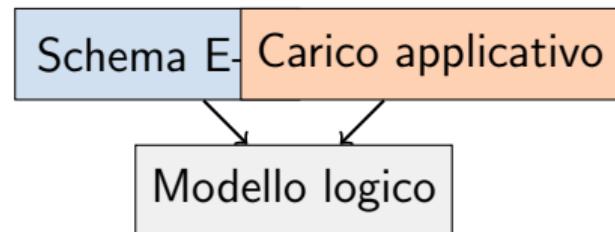
## Ricorda

La progettazione concettuale ha come obiettivo la rappresentazione accurata e naturale dei dati dal punto di vista del *significato*.

# Progettazione Logica: Base per l'Applicazione

La progettazione logica costituisce la **base per l'effettiva realizzazione dell'applicazione** e deve tener conto, per quanto possibile, delle sue **prestazioni**.

Questa necessità può portare a una **ristrutturazione** dello schema concettuale che renda più efficiente l'esecuzione delle operazioni previste.



# Le Due Fasi

È necessario prevedere:

- ① Un'attività di **riorganizzazione**
- ② Un'attività di **traduzione**

La progettazione logica si articola quindi in **due fasi**:

- **Ristrutturazione** dello schema Entità-Relazione
- **Traduzione** verso il modello logico

# Processo di Progettazione



# Output della Progettazione Logica

Costituiscono i prodotti finali:

- **Lo schema logico finale**
- **I vincoli di integrità** definiti su di esso
- **La relativa documentazione**

## Esempio

Musei(CodiceMuseo, Denominazione, Città, Indirizzo, NumeroOpere)

# Il Modello Logico

Il **modello logico** del database (o schema logico) è lo strumento che viene utilizzato come *input* per la **progettazione fisica** dei database.

Deve quindi avere il **massimo livello di dettaglio e precisione possibile**.

## Funzioni del Modello Logico

Dota i dati di una struttura utile per **realizzare, semplificare e ottimizzare** le operazioni di archiviazione, interrogazione e manipolazione dei dati.

# Modello Logico: Contenuti

Deve contenere tutte le informazioni necessarie per **definire fisicamente le tabelle**, riportando la **descrizione puntuale e completa** del significato di ogni dato che viene memorizzato.

## Esempio Schema Relazionale

alunni (matricola(pk), cognome, nome, ..., scuola(fk))

Questo può essere considerato come *schema logico*, ma è sempre preferibile realizzare un **modello logico preciso** (o completo) per rendere più semplice la progettazione fisica.

# Componenti del Modello Logico Preciso

Un modello logico preciso comprende:

- ① Per ogni **entità**, l'elenco completo degli attributi
- ② Per ogni **entità**, l'indicazione esplicita della chiave primaria e di eventuali chiavi alternative
- ③ Per ogni **entità**, l'indicazione esplicita di eventuali chiavi esterne
- ④ Per ogni **attributo**, l'indicazione esplicita di opzionalità o obbligatorietà

# Componenti del Modello Logico Preciso (2)

- ⑤ Per ogni **attributo**, l'indicazione esplicita del tipo di dati (*data type*), che ne specifichi il formato e la lunghezza, con la segnalazione dei valori ammessi
- ⑥ Per ogni **relazione**, l'indicazione della molteplicità minima e massima in entrambe le direzioni
- ⑦ Per ogni **relazione**, l'indicazione delle regole di integrità referenziale applicabili

# Esempio: Modello Logico Preciso

Attributo/campo	Tipo	Dimensione	Valori	Note
Matricola	Numerico	8	Autoincrement	pk
Cognome	Stringa	40	Not null	
Nome	Stringa	30	Not null	
...				
Scuola	Numerico	12	Codifica del MIUR	fk

Durante la progettazione logica si definiscono inoltre gli **schemi esterni (viste)** per le specifiche applicazioni.

# Ristrutturazione: Panoramica

Una volta che lo schema E-R è stato completamente definito, si procede al suo **affinamento** (ristrutturazione) attraverso:

## Regola Base

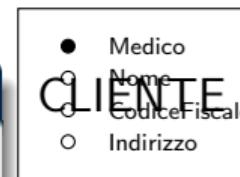
**Eliminazione dei costrutti non rappresentabili** applicando delle regole base di modellizzazione

# Eliminazione Costrutti: Entità Scollegate

Le **entità non possono** essere modellate se sono **scollegate** da altre entità.

## Eccezione

Un database formato da una *singola tabella* (esempio: carrello elettronico nei siti di commercio elettronico)



# Eliminazione Attributi Composti

Nel caso fosse presente un **attributo composto** si può procedere in due modi alternativi:

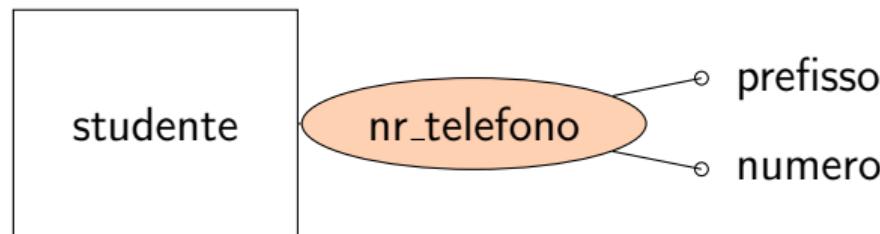
- ① **Considerare** tutti i sottoattributi come attributi
- ② **Eliminare** i sottoattributi e considerare l'attributo composto come un attributo semplice



# Attributi Composti: Soluzione A

## Soluzione A: scomposizione

Nell'entità studente è presente come attributo composto il nr\_telefono.

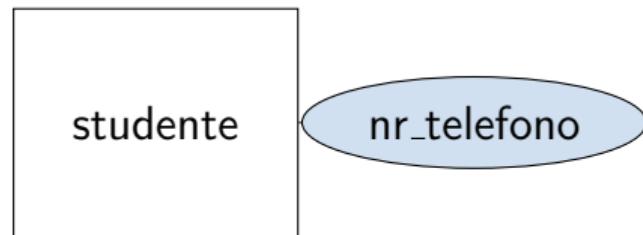


(0, 1)

Si scomponete in: tel\_prefisso e tel\_numero

# Attributi Composti: Soluzione B

## Soluzione B: non considerazione



(0, 1)

Si mantiene come unico attributo: nr\_telefono

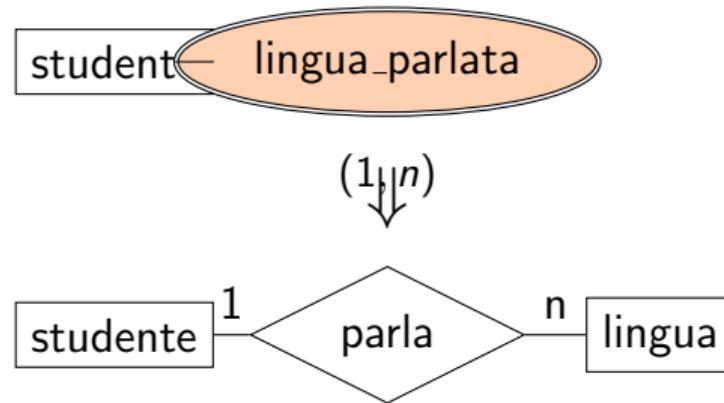
# Eliminazione Attributi Multivalore

Gli eventuali **attributi multivale**ro presenti devono essere "**promossi**" a entità.

Si crea una *nuova entità* che contiene i valori dell'attributo e la si collega all'entità che possedeva l'attributo mediante una nuova relazione **uno-a-molti** o **molti-a-molti**, a seconda dei casi.

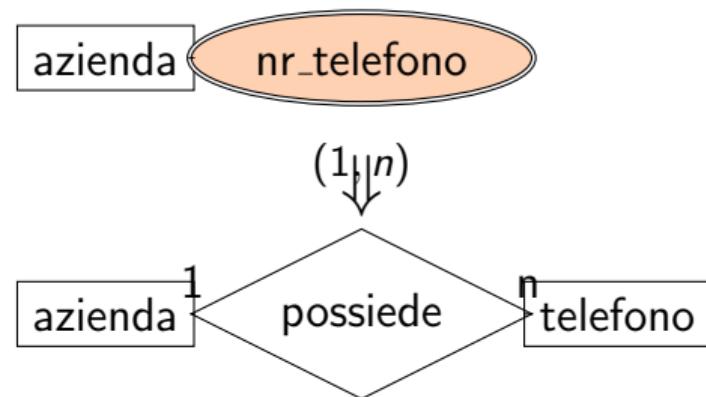
# Attributi Multivalore: Esempio 1

**Esempio:** Uno studente può parlare più di una lingua.

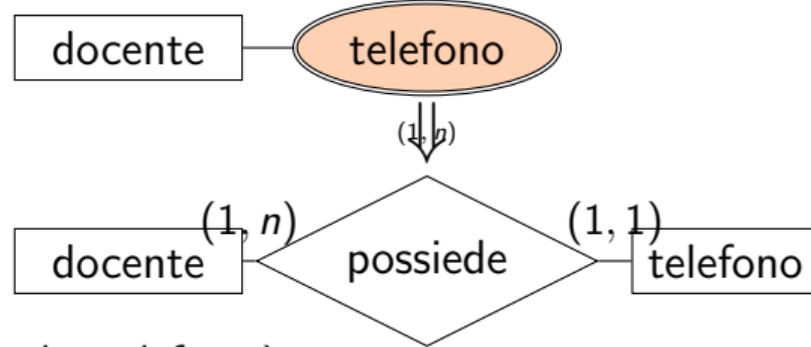


# Attributi Multivalore: Esempio 2

**Esempio:** Un'azienda può possedere più numeri di telefono.



# Attributi Multivalore: Esempio 3



Dato che *telefono* è un attributo multivalore, è necessario introdurre una nuova entità nella fase di ristrutturazione.

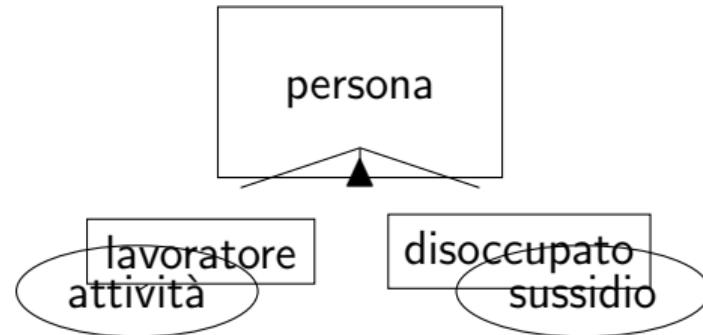
# Eliminazione Gerarchie e Specializzazioni

La **gerarchia di generalizzazione** deve essere trasformata in una unica entità in quanto non è supportata dal modello relazionale.

È possibile ristrutturare il diagramma in due modi differenti:

- ① **Eliminazione dei figli** (collasso dei figli), aggiungendo uno o più attributi nell'entità padre
- ② **Eliminazione del padre** (esplosione dei figli): completando ciascun figlio con gli attributi del padre

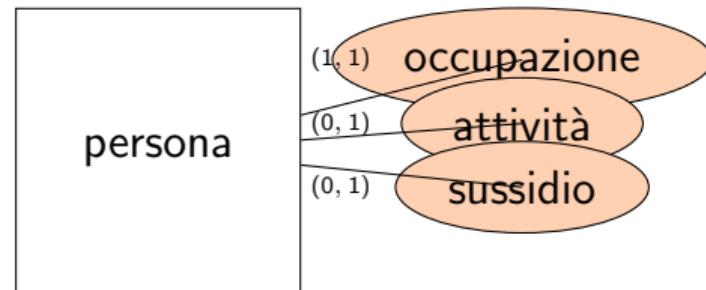
# Gerarchia: Esempio



Le entità *lavoratore* e *disoccupato* sono figlie dell'entità *persona*.

# Soluzione A: Collasso dei Figli

**Caso A:** Una sola entità nella quale sono stati aggiunti gli attributi che rimarranno vuoti a seconda del caso della singola istanza.



# Soluzione B: Esplosione dei Figli

**Caso B:** Due entità distinte.



Nel caso A abbiamo una sola entità nella quale sono stati aggiunti gli attributi che rimarranno vuoti a seconda del caso della singola istanza; nel caso B abbiamo invece due entità distinte.

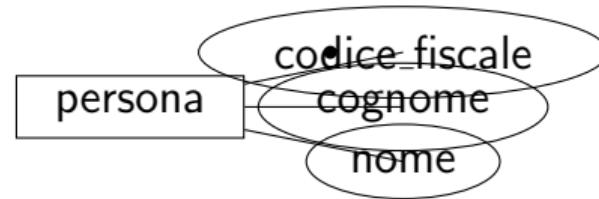
# Traduzione: Panoramica

Dopo aver applicato al modello E-R le regole di ristrutturazione si ottiene lo **schema E-R ristrutturato**, pronto per essere trasformato nello schema relazionale.

In questa fase, che prende il nome di **fase di traduzione**, vengono applicate le regole di trasformazione di entità, attributi e associazioni dello schema E-R in relazioni del modello relazionale.

# Traduzione delle Entità

Per ogni **entità** viene generata una **tabella** indicando il suo *schema relazionale*, dove viene riportato un **campo** per ogni **attributo** dell'entità.



## Schema Relazionale

persona (codice\_fiscale(pk), cognome, nome)

# Attributi Calcolati

In caso di presenza di **attributi calcolati** va aggiunto un vincolo di integrità che specifica il metodo di calcolo.

## Example

Un caso frequente è l'attributo età: questo viene calcolato come differenza di due date (data odierna e data di nascita), e se manca il valore di una di esse il suo valore deve essere NULL.

# Trasformazione delle Relazioni

Si effettua la traduzione delle relazioni, in base a:

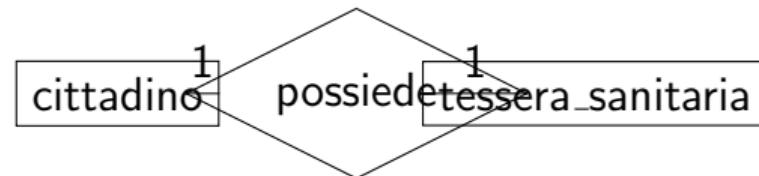
- Numero di entità partecipanti
- Loro cardinalità

Tipologie:

- ① Relazioni uno-a-uno (1,1)
- ② Relazioni uno-a-molti (1,N)
- ③ Relazioni molti-a-molti (N,N)
- ④ Relazioni ternarie

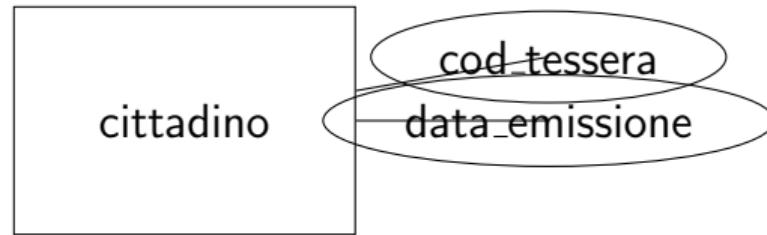
## Relazioni Uno-a-Uno (1,1)

Due entità legate da una relazione (1,1) possono essere **ridotte a un'unica entità** che contiene gli attributi sia della prima sia della seconda entità.



La relazione è del tipo 1,1, in quanto un cittadino può possedere una sola tessera sanitaria e ogni tessera sanitaria è unica per ogni cittadino: quindi la relazione è eliminabile e otteniamo un'unica entità con tutti gli attributi.

# Relazioni Uno-a-Uno: Risultato



Gli attributi della tessera sanitaria vengono incorporati nell'entità cittadino.

# Relazioni Uno-a-Molti (1,N)

Ogni associazione **uno a molti** può essere tradotta inserendo un **attributo aggiuntivo** sulla entità da cui la cardinalità massima è 1.

Tale attributo conterrà la *chiave dell'oggetto associato* all'oggetto corrente. Si introduce così una **chiave esterna (foreign key)** e, con essa, un vincolo di integrità referenziale.

# Relazioni 1-N: Esempio



## Schema Relazionale

PROVINCIA(NomeP, Persona)

PERSONA(CF, Nome, NomeP, Dal)

FK: NomeP REFERENZIA PROVINCIA

# Relazioni Multi-a-Multi (N,N)

Le relazioni **N,N** non possono essere rappresentate nel modello logico relazionale.

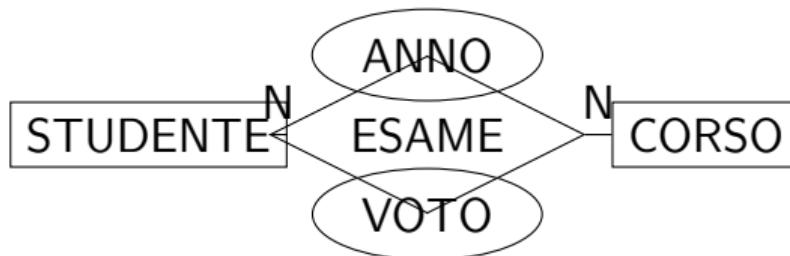
Devono essere tradotte aggiungendo un'**entità associativa** (anche detta entità "ponte") che va messa in relazione con le due entità originali.

# Relazioni N-N: Traduzione

Una relazione molti a molti che lega due entità si traduce in:

- Una **tabella per ogni entità** contenente gli attributi dell'entità
- Una **tabella per la relazione** contenente:
  - Gli attributi identificatori delle entità che collega
  - Gli attributi propri
- Gli attributi identificatori della tabella collegata alla relazione compongono la **chiave esterna**

# Relazioni N-N: Esempio Studente-Corso



## Schema Relazionale

Studente(Matricola, Nome)

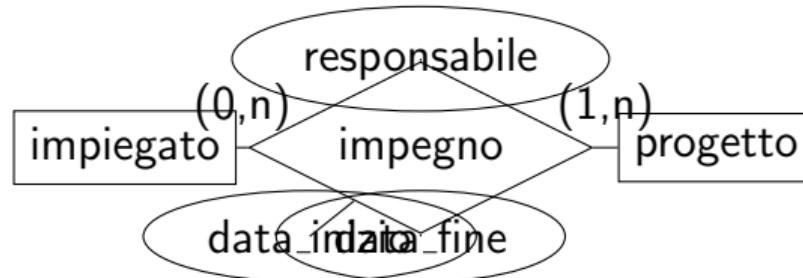
Corso(Codice, Denominazione)

Esame(Matricola, Codice, Anno, Voto)

FK: Matricola REFERENZIA Studente

FK: Codice REFERENZIA Corso

# Relazioni N-N: Esempio Impiegato-Progetto



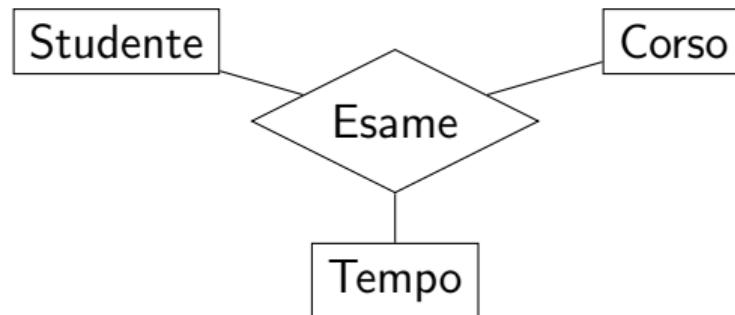
Gli attributi della relazione vengono inseriti nella tabella associativa.

## Schema Relazionale

```
impiegato (ID_impiegato(pk), cognome, nome)
progetto (ID_progetto(pk), descrizione)
impegno (ID_impiegato(pk), ID_progetto(pk), data_inizio, responsabile)
```

# Relazioni Ternarie

Una **relazione ternaria** collega logicamente tre entità.



## Example

Uno studente può ripetere lo stesso esame in tempi diversi.

# Relazioni Ternarie: Caratteristiche

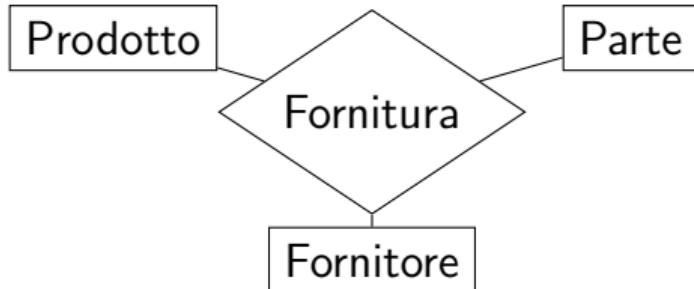
Le cardinalità minime raramente sono 1 per tutte le entità coinvolte in una relazione ternaria.

Le cardinalità massime di una relazione n-aria sono (praticamente) sempre **N**.

## Nota importante

Se la partecipazione di un'entità E ha cardinalità massima 1, è possibile eliminare la relazione n-aria e legare l'entità E con le altre mediante relazioni binarie.

# Relazioni Ternarie: Esempio Fornitura



- Il venditore A fornisce stampanti al Dipartimento Personale
- Il venditore B fornisce fotocopiatrici al Dipartimento Ricerca

# Relazioni Ternarie: Traduzione

La relazione ternaria con cardinalità molti a molti viene trasformata nel seguente modo:

- Ogni **entità** in una tabella
- La **relazione** in una tabella contenente:
  - Tutti gli attributi chiave delle entità collegate
  - L'attributo proprio (se presente)

# Relazioni Ternarie: Schema Finale

## Schema Relazionale

Prodotto(CodProd, Nome)

Fornitore(CodForn, Telefono)

Parte(CodPart, Descrizione)

Fornitura(CodPart, CodProd, CodForn, Quantità)

FK: CodPart REFERENZIA Parte

FK: CodProd REFERENZIA Prodotto

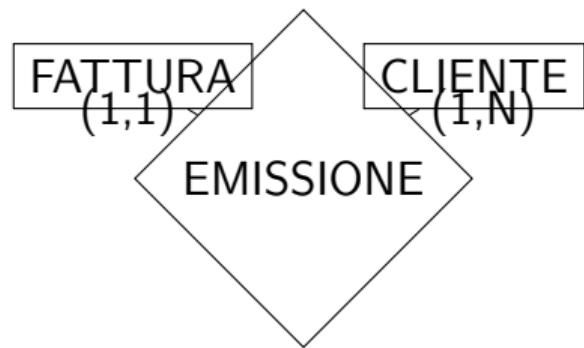
FK: CodForn REFERENZIA Fornitore

# Esempio Complesso: Fattura-Cliente-Prodotto



La relazione ternaria con una cardinalità (1,1) può essere scomposta in relazioni binarie.

# Scomposizione Relazione Ternaria



# Eliminare le Relazioni Ridondanti

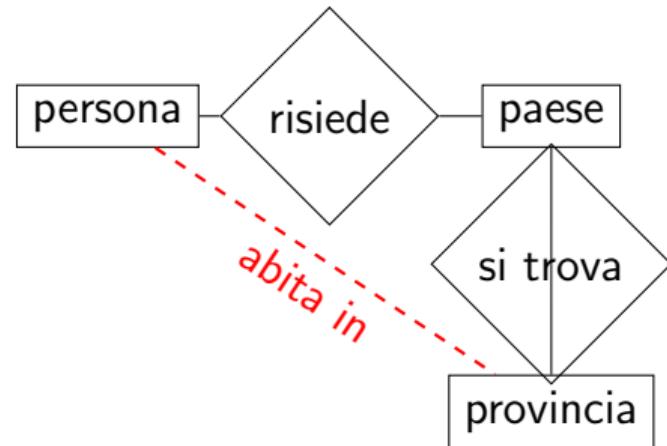
Una **relazione ridondante** è una relazione definita tra due entità e che è equivalente nel significato a un'altra relazione tra le stesse due entità che passa attraverso un'entità intermedia.

## Example

- Una persona abita in una provincia
- Una persona risiede in un paese
- Il paese si trova in una provincia

La prima relazione è **ridondante** rispetto alle altre due.

# Ridondanza: Esempio Visivo



La relazione tratteggiata è **eliminabile**.

# Definizione delle Chiavi

Aggiungere anche la definizione delle chiavi:

- Le **chiavi artificiali** iniziano con ID
- Per convenzione usiamo ID *maiuscolo* per la chiave primaria
- Indichiamo con id *minuscolo* le chiavi esterne

## Example

```
STUDENTE(ID_studente(PK), nome, cognome, id_classe(FK))
CLASSE(ID_classe(PK), sezione, anno)
```

# Vincoli da Considerare

- ① Vincoli **NOT NULL** per gli attributi obbligatori
- ② Vincoli di **interdipendenza** di valori nulli
- ③ Vincoli di **chiave** (primarie e non)
- ④ Vincoli di **foreign key** che provengono dalla traduzione di relazioni
- ⑤ Vincoli di **generalizzazione**, formulati come vincoli insiemistici

## Partecipazione obbligatoria (cardinalità minima 1)

Diventa vincolo di inclusione o foreign key dalla relazione che corrisponde all'entità verso quella che corrisponde alla ER-relazione

## Funzionalità (cardinalità massima 1)

Diventa vincolo di chiave sulla relazione che corrisponde alla ER-relazione

# Grazie per l'attenzione!

Domande?