# Basi di dati

UniVR - Dipartimento di Informatica

**Fabio Irimie**

1° Semestre 2025/2026

# Indice

# 1 Introduzione

Le basi di dati sono raccolte di dati strutturati, organizzati in modo tale da permettere un facile accesso. Questi dati sono persistenti, ovvero rimangono memorizzati anche dopo la chiusura del programma che li ha creati.

# 2 Sistema informativo

Un sistema informativo è l'insieme delle attività umane e dei dispositivi di memorizzazione ed elaborazione che organizza e gestisce l'informazione di interesse per un organizzazione di dimensioni qualsiasi. Non contiene necessariamente dati memorizzati in un computer.

Un sistema informativo è composto da:

- **Dato**: è l'elemento di conoscenza di base costituito da simboli che devono essere elaborati
- **Informazione**: è l'interpretazione dei dati che permette di ottenere una conoscenza più o meno esatta di fatti e situazioni

## 2.1 Base di dati

**Definizione 2.1.** Una **base di dati** è una **collezione di dati persistenei** utilizzati per rappresentare **con tecnologia informatica** le informazioni di interesse per un **sistema informativo**

La soluzione convenzionale per la gestione dei dati è l'uso di file, ma questa presenta alcuni problemi:

- Scarsa efficienza nell'accesso ai dati (accesso sequenziale)
- Ridondanza nei dati
- Inconsistenza nei dati (aggiornamenti parziali)
- Progettazione dei dati replicata per ogni applicazione

Per risolvere questi problemi si è creato un livello di astrazione maggiore tra le applicazioni e il filesystem, ovvero il **Data Base Management System (DBMS)**.

**Definizione 2.2.** Un **DBMS** è un sistema che gestisce su **memoria secondaria** collezioni di dati (chiamate "basi di dati"). Le caratteristiche principali sono:

- Grandi
- Condivise, cioè accessibili da più utenti
- Persistenti

Un DBMS assicura:

- Affidabilità, cioè nessuna perdita di dati
- Privatezza
- Accesso efficiente

### 2.1.1 Modello dei dati

Un **modello dei dati** è un insieme di strutture che permettono di descrivere una base di dati. Per accedere a questi dati si usano delle **interrogazioni**, cioè delle richieste, in un linguaggio dichiarativo specifico, che permettono di ottenere i dati desiderati.

Ci sono diversi linguaggi per interagire con un DBMS:

- Linguaggio per la definizione dei dati (DDL), consente di definire la struttura della base di dati
- Linguaggio per l'interrogazione e aggiornamento dei dati (DML), consente di interrogare e aggiornare i dati
  - Linguaggio di interrogazione: estrae informazioni da una base di dati, ad esempio SQL, algèbre relazionale, calcolo relazionale
  - Linguaggio di manipolazione: popola la base di dati, modifica il suo contenuto con aggiunge, cancellazioni e variazioni sui dati, ad esempio SQL

Il modello di dati è un insisme di **costrutti** forniti dal DBMS per descrivere la struttura e le proprietà dell'informazione contenute in una base di dati.

Ci sono diversi tipi di modelli di dati:

- **Modelli di dati del passato**:
  - Modello reticolare
  - modello gerarchico
- **Modelli di dati attuali**:
  - Modello relazionale
  - Modello ad oggetti
  - Modello a oggetti-relazionale
  - Modello basato su documenti (JSON)
  - Modelli NoSQL

I modelli vengono utilizzati per creare:

- **Schema di una base di dati**: è la descrizione della struttura e delle proprietà di una specifica base di dati fatta utilizzando i costrutti del modello dei dati (lo schema di una base di dati è invariante nel tempo)
- **Istanza di una base di dati**: è costituita dai **valori effettivi** che in un certo istante popolano le strutture dati (l'istanza di una base di dati varia nel tempo)

Lo schema di una base di dati è diviso in tre livelli:

- **Schema esterno**: è la visione dell'utente della base di dati, cioè la parte di base di dati che interessa a un particolare utente o gruppo di utenti
- **Schema logico**: è la visione globale della base di dati, cioè la struttura logica della base di dati che descrive tutti i dati e le relazioni tra essi
- **Schema interno**: è la rappresentazione fisica della base di dati, cioè il modo in cui i dati sono effettivamente memorizzati nella memoria secondaria

Le proprietà dello schema sono:

- **Indipendenza fisica**: lo schema logico della base di dati è completamente indipendente dallo schema interno
- **Indipendenza logica**: gli schemi esterni della base di dati sono indipendenti dallo schema logico

# 3 Progettazione di una base di dati

Il ciclo di vita di un processo di automazione di un sistema informativo è diviso in diverse fasi:

- **Studio di fattibilità**: si valuta se l'automazione del sistema informativo è possibile e conveniente
- **Raccolta e analisi dei requisiti**: si individuano proprietà e funzionalità del sistema (dati e applicazioni) producendo una descrizione completa ma informale
- **Progettazione**: si produce una descrizione formale del sistema informativo

La progettazione si divide in due parti principali che vanno di pari passo:

- **Progettazione dei dati**: si produce una descrizione formale dei dati (schema). Una volta progettati i dati vengono implementati in un DBMS
- **Progettazione delle applicazioni**: si produce una descrizione formale delle applicazioni (specifica)

Una volta implementati i dati e le applicazioni si passa alla fase di **validazione e collaudo**

## 3.1 Progettazione dei dati

Una metodologia di progettazione dei dati è costituita da:

- **Decomposizione**: dividere in passi le attività di progetto
- **Strategie**: individuare un insieme di strategie e criteri di scelta da seguire
- **Modelli di riferimento**: utilizzare modelli di dati e tecniche di progettazione consolidate

Una buona metodologia deve essere:

- Generale
- Facile da usare
- Deve produrre un risultato di qualità

## 3.2 Requisiti

### 3.2.1 Progettazione concettuale

La progettazione concettuale è la prima fase della progettazione dei dati. Lo scopo è quello di produrre una descrizione formale dei dati (schema concettuale). Lo schema deve essere **indipendente dall'implementazione**.

Non è solo un progetto intermedio, ma costituisce anche una porzione del risultato finale perchè rappresenta una descrizione di **alto livello** del contenuto della base di dati, comprensibile anche per utenti poco esperti.

### 3.2.2 Progettazione logica

La progettazione logica è la seconda fase della progettazione dei dati. Lo scopo è quello di tradurre lo schema concettuale in uno schema logico in modo da poterlo utilizzare su un sistema specifico. Lo schema logico infatti è dipendente dalle tecnologie utilizzate. Bisogna tenere anche in considerazione le operazioni più frequenti che le applicazioni effettueranno sulla base di dati.

### 3.2.3 Progettazione fisica

La progettazione fisica è la terza fase della progettazione dei dati. L'obiettivo è quello di ottimizzare l'accesso ai dati completando lo schema logico con i parametri relativi alla memorizzazione fisica dei dati e con gli opportuni metodi d'accesso (**indici**).

## 3.3 Strategie di progettazione

Lo sviluppo do uno schema concettuale può essere visto come un processo di ingegnerizzazione ed è quindi possibile applicare anche a tale processo le strategie classiche:

- **Top-Down**: consiste nel considerare le specifiche **globalmente** e produrre uno schema iniziale completo ma con **pochi concetti** molto astratti. Si va poi a **raffinare** i concetti astratti fino ad arrivare allo schema concettuale completo in ogni dettaglio
- **Bottom-Up**: consiste nel **decomporre** le specifiche iniziali in parti elementari e produrre uno schema iniziale dettagliato
- **Inside-Out**: consiste nell'individuare nelle specifiche alcuni concetti importanti (detti concetti guida) e partendo da quelli generare gli schemi per i relativi concetti. Si procede fondendo gli schemi precedenti per generare lo schema finale.

### 3.3.1 Analisi di qualità dello shema

L'analisi della qualità dello schema concettuale prodotto può essere suddivisa in diverse fasi:

- **Verifica della correttezza**: Uno schema concettuale è corretto se utilizza correttamente i costrutti del modello concettuale adottato (nel nostro caso E-R). I possibili errori sono:
  - Errori sintattici: si verificano quando i costrutti vengono usati senza seguire le regole sintattiche
  - Errori semantici: si verificano quando i costrutti vengono usati senza seguire le regole semantiche, cioè la loro definizione
- **Verifica della completezza**: Uno schema concettuale è completo se rappresenta tutti i requisiti espressi nelle specifiche.
- **Verifica di minimalità**: Uno schema è minimale quando tutti i concetti descritti nei requisiti sono rappresentati nello schema **una volta sola**. Uno schema non minimale contiene delle **ridondanze**. I casi di probabile ridondanza sono:
  - Ereditarietà nelle relazioni: si manifesta quando due entità padri sono in relazione e le loro entità figlie sono anch'esse in relazione. Per far si che non ci sia ridondanza in questo caso bisogna verificare che le due relazioni non rappresentino lo stesso concetto
  - Cicli di relazioni: si manifesta quando esistono dei cicli tra le relazioni di più entità. Non è detto che ci sia sicuramente ridondanza, ma per evitarla bisogna verificare se la relazione $R_i$ si possa ottenere dalla composizione delle altre relazioni. Se è così allora $R_i$ va eliminata in quanto ridondante.
- **Verifica di leggibilità**: Uno schema concettuale è leggiible quando rappresenta tutti i requisiti in modo naturale e facilmente compresnsibile. Alcuni consigli per migliorare la leggibilità sono:
  - Curare la scelta dei nomi delle componenti dello schema
  - Disegnare bene lo schema

  Si può migliorare la leggibilità anche semplificando lo schema. Le relazioni ternarie non sono facili da leggere e interpretare, quindi solitamente la loro eliminazione può comportare una migliore leggibilità dello schema. Le operazioni possibili sono:
  - Trasformare una relazione ternaria in un'entità (sempre applicabile)
  - Riduzione di una relazione ternariaa a due relazioni binarie (**precondizione**: una delle entità partecipa alla relazione ternaria con cardinalità (1,1) (spiegata nel capitolo 4.5))
  - Trasformare una relazione sovrapposta in una generalizzazione esclusiva esplicitando l'entità che rappresenta la sovrapposizione

# 4 Progettazione Concettuale (Modello Entità - Relazione)

È un modello, formale e non ambiguo, utilizzato per la progettazione concettuale di una base di dati. Fornisce strumenti formali (costrutti), con sintassi grafica, per specificare la struttura e le proprietà dei dati da rappresentare indipendentemente dalla tecnologia.

Ogni costrutto viene definito specificando:

- Il suo significato (o semantica)
- La sua sintassi grafica
- La rappresentazione delle sue istanze (o occorrenze)

Progettare indipendentemente dalle tecnologie significa:

- **Non considerare** eventuali ottimizzazioni
- **Considerare** tutti i requisiti senza semplificazioni o convenzioni
- **Considerare** sempre i processi di generazione e modifica dei dati per verificare che ogni situazione sia rappresentabile da un'istanza "pulita" della base di dati

## 4.1 Entità

Un'entità E rappresenta una **classe di oggetti** che hanno le seguenti caratteristiche:

- **Proprietà comuni**
- **Eistenza autonoma** rispetto ad altre classi di oggetti
- **Identificazione univoca**, cioè esiste una chiara corrispondenza tra gli oggetti istanze di entità e concetti istanziati nel sistema informativo

Un entità si rappresenta con un rettangolo che contiene il nome dell'entità:


Figura 1: Rappresentazione grafica di un'entità

### 4.1.1 Istanza

Un'istanza dell'entità E è un **oggetto** appartenente alla classe rappresentata da E. Si indica con $\mathrm{I}(\mathrm{E})$ l'insieme delle istanze di E che esistono nella base di dati in un certo istante e alla creazione della base di dati è vuota: $\mathrm{I}(\mathrm{E}) = \emptyset$.

**Esempio 4.1.** Rappresentiamo con il costrutto entità il concetto di **persona**. Bisogna gestire nella base di dati le informazioni che descrivono un gruppo di persone.


Figura 2: Rappresentazione grafica dell'entità Persona

L'insieme delle istanze dell'entità Persona è il seguente:

$$I(\text{Persona}) = \{p_1, p_2, p_3, \ldots\}$$

## 4.2 Relazione

Una relazione R rappresenta un **legame logico** tra **due o più** entità. Può esserci anche una relazione all'entità stessa (relazione ricorsiva).

Una relazione si rappresenta nello schema con un rombo a cui si collegano attraverso delle linee le entità coinvolte nella relazione. Il nome della relazione viene scritto a fianco al rombo:

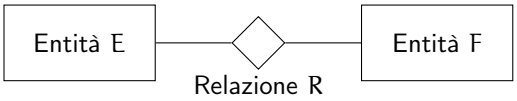


Figura 3: Rappresentazione grafica di una relazione tra due entità

### 4.2.1 Istanza

Data una relazione R tra $n$ entità $E_1, \ldots, E_n$ un'istanza della relazione R è una **ennupla di istanze di entità**:

$$(e_1, \ldots, e_n) \text{ dove } e_i \in I(E_i) \text{ per } 1 \leqslant i \leqslant n$$

La popolazione di R rappresenta l'insieme delle coppie di istanze delle entità E e F che sono in relazione in un certo istante:

$$I(R) = \{(e_i, f_j) \mid e_i \in I(E), f_j \in I(F)\}$$

**Esempio 4.2.** Supponiamo che nello schema ci siano le entità **Persona** e **Comune**, bisogna gestire la **Residenza** delle persone nei comuni italiani.

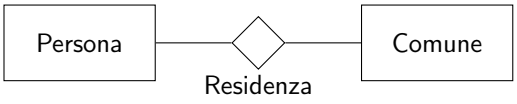


Figura 4: Rappresentazione grafica della relazione Residenza

Ciò implica che per esistere un'istanza di residenza devono esistere un'istanza

di persona e un'istanza di comune.

Data una relazione R tra $n$ entità $\{E_1, E_2, \ldots, E_n\}$ vale **sempre** la seguente proprietà sull'insieme delle istanze $I(R)$:

$$I(R) \subseteq I(E_1) \times I(E_2) \times \ldots \times I(E_n)$$

La conseguenza di questa proprietà è che non è possibilie rappresentare la stessa ennupla più volte.

### 4.2.2 Relazione ricorsiva

È una relazione binaria sulla stessa entità:

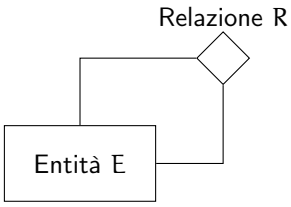


Figura 5: Rappresentazione grafica di una relazione ricorsiva

## 4.3 Attributo

Rappresenta una proprietà elementare di un'entità o di una relazione. Ogni attributo di un'entità o di una relazione associa ad ogni istanza **un solo** valore appartenente ad un dominio di valori ammissibili. Può essere visto come una funzione che ha come dominio le istanze dell'entità (o relazione) e come codominio l'insieme dei valori ammissibili:

$$f_A : I(E) \mapsto D$$

dove $a$ è un attributo dell'entità $E$, mentre $I(E)$ l'insieme delle istanze di $E$ e $D$ è l'insieme dei valori ammissibili.

La sintassi grafica di un attributo è un cerchio **vuoto** collegato con una linea all'entità con accanto il nome dell'attributo:


Figura 6: Rappresentazione grafica di un attributo di un'entità

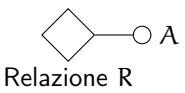


Figura 7: Rappresentazione grafica di un attributo di una relazione

**Esempio 4.3.** Rappresentiamo il concetto di persona tramite un'entità, bisogna gestire nella base di dati il nome e il cognome di un gruppo di persone.

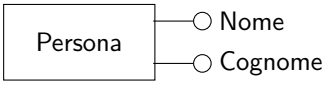


Figura 8: Rappresentazione grafica dell'entità Persona con gli attributi Nome e Cognome

### 4.3.1 Attributo opzionale e multivalore

L'attributo opzionale o multivalore si ottiene da un attributo normale specificando un vincolo di cardinalità (spiegato più avanti per le relazioni 4.5) sui valori che l'attributo può assumere (il default è (1,1)). I valori possibili sono i seguenti:

- (0,1): attributo opzionale
- (1,N): attributo multivalore obbligatorio
- (0,N): attributo multivalore opzionale

La sintassi grafica per rappresentare un attributo opzionale o multivalore è la seguente:

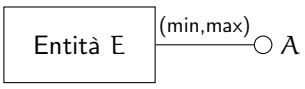


Figura 9: Rappresentazione grafica di un attributo opzionale o multivalore

### 4.3.2 Attributo composto

Permette di raggruppare gli attributi di un'entità o di una relazione che hanno un significato comune.

La sintassi grafica per rappresentare un attributo composto è la seguente:

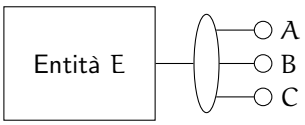


Figura 10: Rappresentazione grafica di un attributo composto

## 4.4 Identificatore

Data un'entità E, un identificatore è un insieme di proprietà (attributi e/o relazioni) che **identificano univocamente** ogni istanza di E. Un insieme di proprietà identifica univocamente le istanze di un'entità E se **non esistono** due istanze di E che presentano gli stessi valori o istanze nelle proprietà dell'insieme.

La sintassi grafica di un identificatore per un'entità con un solo attributo $a$ è la seguente:


Figura 11: Rappresentazione grafica di un identificatore con un solo attributo

Per un identificatore costituito da più attributi $a, b$ la sintassi grafica è la seguente:

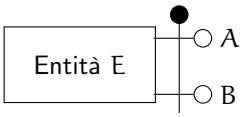


Figura 12: Rappresentazione grafica di un identificatore con più attributi

Può anche esistere un identificatore costituito da una relazione R (deve essere una funzione):

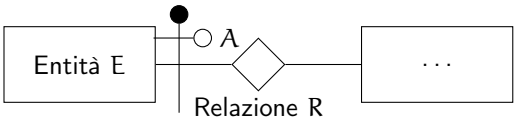


Figura 13: Rappresentazione grafica di un identificatore costituito da un attributo e una relazione

Oppure un identificatore può essere costituito soltanto da una relazione R:

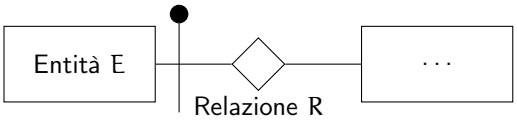


Figura 14: Rappresentazione grafica di un identificatore costituito da una relazione

Ci sono due tipi di identificatori:

- **Identificatori interni**: sono costituiti solo da attributi dell'entità
- **Identificatori esterni**: sono costituiti da almeno una relazione con un'altra entità

**Attenzione**: Non esistono identificatori sulle relazioni perchè ogni relazione è già univocamente identificata dalle istanze delle entità coinvolte.

**Esempio 4.4.** Rappresentiamo il concetto di persona tramite un'entità, bisogna gestire nella base di dati il codice fiscale, il nome, il cognome e la data di nascita di un gruppo di persone

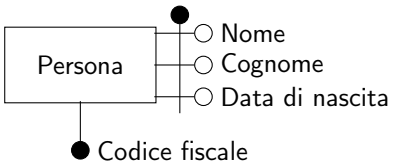


Figura 15: Esempio di identificatore interno

**Definizione 4.1.** Un **vincolo di identificazione** limita la popolazione di un'entità impedendo l'esistenza di due istanze con gli stessi valori nelle proprietà che costituiscono l'identificatore.

La scelta dell'identificatore va sempre fatta considerando le proprietà significative per il sistema informativo. A **livello concettuale** è quindi da **evitare** l'introduzione di nuovi attributi identificatori (ad esempio l'ID).

## 4.5 Cardinalità

Tra due relazioni esistono dei vincoli di cardinalità che limitano il numero di istanze di una entità che possono essere in relazione con una singola istanza dell'altra entità. Data una relazione R i vincoli di cardinalità vengono specificati per ogni entità $E_i$ coinvolta nella relazione R e specificano il numero **minimo** e **massimo** di istanze di R a cui un istanza di $E_j$ deve o può partecipare.

La sintassi grafica per rappresentare i vincoli di cardinalità è la seguente:

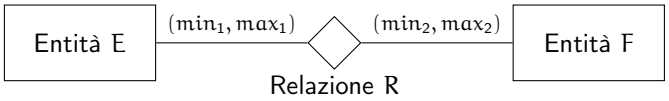


Figura 16: Rappresentazione grafica dei vincoli di cardinalità

### 4.5.1 Valori possibili per $MIN_i$

I possibili valori che il minimo $MIN_i$ può assumere sono:

- 0: Indica che la partecipazione alla relazione R delle istanze di $E_i$ è **opzionale**
- 1: Indica che la partecipazione alla relazione R delle istanze di $E_i$ è **obbligatoria**
- $num > 1$: Indica che per ogni istanza di $E_i$ devono essere presenti almeno $num$ occorrenze della relazione R che la coinvolgono

### 4.5.2 Valori possibili per $MAX_i$

- 1: Indica che un'istanza di $E_i$ può **al massimo** partecipare a una sola occorrenza della relazione R (se R è binaria questo indica che R **è una funzione**)

- N: Indica che un'istanza di $E_i$ può partecipare a più occorrenze della relazione R senza limite massimo
- $num > 1$: Indica che per ogni istanza di $E_i$ possono essere presenti **al massimo** $num$ occorrenze della relazione R che la coinvolgono

**Esempio 4.5.** Rappresentiamo il concetto di persona e comune tramite due entità, bisogna gestire la residenza delle persone nei comuni italiani. Questi requisiti non rappresentano la realtà perchè una persona deve avere obbligatoriamente una residenza e può avere al massimo una residenza, mentre un comune può avere zero o più persone residenti.

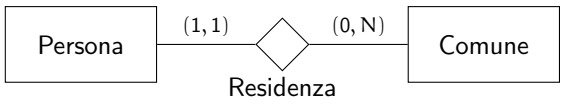


Figura 17: Esempio di Residenza con i vincoli di cardinalità

**Definizione 4.2.** Una relazione R può partecipare ad un identificatore esterno di un'entità E **solo se** tale entità partecipa alla relazione R con vincolo di cardinalità $(1,1)$, quindi se R è una **funzione** che associa ad ogni istanza di E una e una sola istanza dell'altra entità coinvolta in R. Questo vale anche quando R è la sola proprietà che partecipa all'identificatore.

Si potrebbe utilizzare anche il diagramma UML per rappresentare i concetti del modello Entità-Relazione, ma questa rappresentazione segue regole diverse, ad esempio la posizione dove specificare nel diagramma UML i vincoli di cardinalità è invertita rispetto all'ER.

## 4.6 Generalizzazione di entità

La generalizzazione è un legame logico (simile ad un'ereditarietà tra classi) tra un'entità padre E e $n$ $(n > 0)$ entità figlie $E_1, E_2, \ldots, E_n$ dove E rappresenta una classe di oggetti più generale rispetto alle classi di oggetti rappresentate dalle entità figlie.

La sintassi grafica per rappresentare una generalizzazione è la seguente:

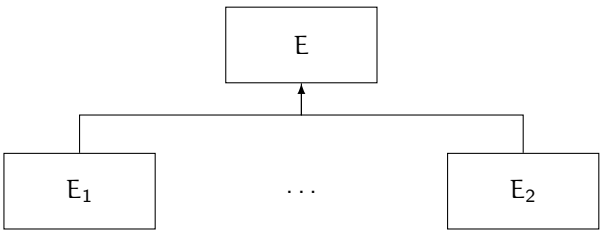


Figura 18: Esempio di Residenza con i vincoli di cardinalità

### 4.6.1 Proprietà delle istanze generalizzate

- Ogni istanza di un'entità figlia $E_i$ è anche istanza dell'entità padre E
- Ogni proprietà (attributi, identificatori e relazioni) dell'entità padre E è anche proprietà di ogni entità figlia $E_i$

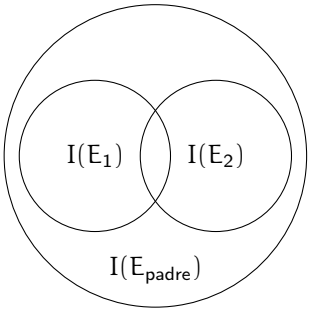


Figura 19: Istanze di entità generalizzate

### 4.6.2 Classificazione delle generalizzazioni

Una generalizzazione può essere classificata secondo due criteri:

- **Totale** se ogni istanza dell'entità padre E è anche istanza di **almeno una** delle entità figlie $E_i$, altrimenti si dice **parziale**
- **Esclusiva** se ogni istanza dell'entità padre E è anche istanza di **al massimo una** delle entità figlie $E_i$, altrimenti si dice **sovrapposta**

Per indicare la classificazione di una generalizzazione si mettono le seguenti notazioni accanto alla freccia della generalizzazione:

- (p,s): parziale e sovrapposta:
- (p,e): parziale ed esclusiva: non si possono avere istanze comuni tra le entità figlie
- (t,s): totale e sovrapposta: non si possono avere istanze dell'entità padre che non siano istanze di almeno una delle entità figlie
- (t,e): totale ed esclusiva: ogni istanza dell'entità padre è istanza di **esattamente una** delle entità figlie

### 4.6.3 Relazioni ternarie

Una relazione ternaria è una relazione che coinvolge tre entità e si usa quando:

- Un'istanza del concetto per esistere nel sistema informativo richiede sempre la presenza di tre istanze di entità (una per ogni entità coinvolta nella relazione)
- Non esistono eccezioni al punto precedente

- Non esistono situazioni nelle quali la terna di entità debba essere rappresentata più volte

La sintassi grafica per rappresentare una relazione ternaria è la seguente (i vincoli di cardinalità sono richiesti, ma qua sono omessi):

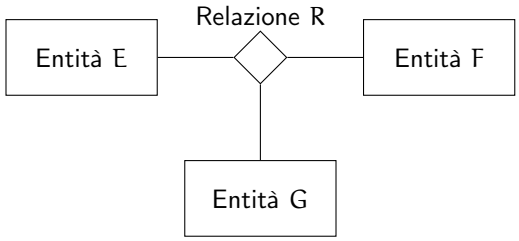


Figura 20: Rappresentazione grafica di una relazione tra due entità

Un uso errato avviene quando per rappresentare lo stato del sistema informativo sarebbe necessario che un'istanza della relazione **contenga solo una coppia e non una terna** di istanze di entità (cioè quando manca un attore). Quindi in questo caso si può sostituire la relazione ternaria con 3 relazioni binarie indipendenti.

L'utilizzo delle relazioni ternarie è utile per "storicizzare" le relazioni binarie. Nella rappresentazione di una relazione binaria storicizzata è necessario poter rappresentare più volte **la stessa coppia in tempi diversi**. Pertanto la relazione binaria può essere trasformata in una relazione ternaria includendo l'entità **Tempo** come nuovo attore nella relazione in modo da poter rappresentare più volte la stessa coppia in istanti di tempo diversi.

# 5 Implementazione dei dati (Modello Relazionale)

Il modello relazionale è un modello dei dati, quindi permette di definire le proprietà che la popolazione di una base di dati deve rispettare. I costrutti principali del modello relazionale sono:

- **Domini di base**: insieme di valori atomici (non scomponibili) che rappresentano i tipi di dati elementari
- **Relazione (o tabella)**: rappresenta un insieme di entità o di relazioni tra entità. Sono definite come un insieme di ennuple o tuple.
- **Superchiavi, chiavi e chiavi primarie**: insieme di attributi che identificano univocamente ogni ennupla di una relazione
- **Vincoli di integrità referenziale**: vincoli che garantiscono la coerenza tra i dati memorizzati in relazioni diverse
- **Vincoli di integrità generici**: vincoli che garantiscono la coerenza dei dati memorizzati in una singola relazione

## 5.1 Domini di base

Sono i domini da cui si scelgono i valori delle proprietà delle istanze di informazione da rappresentare. I domini tipici sono:

- Caratteri
- Stringhe di caratteri
- Numeri interi
- Numeri decimali a virgola fissa
- Numeri decimali a virgola mobile
- Domini del tempo, per rappresentare istanti e intervalli di tempo
- ecc...

I domini presenti variano a seconda del sistema, ma in questo corso vedremo il linguagigo SQL nel sistema postgreSQL.

## 5.2 Relazione

Una relazione può essere vista come una tabella.

**Esempio 5.1.** Ad esempio:

| | | |
|---|---|---|
| Milano | 20100 | 1.300.000 |
| Verona | 37100 | 350.000 |
| Brescia | 25100 | 250.000 |

Tabella 1: Esempio di tabella che rappresenta alcune città italiane

Una tabella è un contenitore di dati la cui struttura è caratterizzata da una lista di colonne:

- I dati sono scritti nelle righe dove **ogni riga** descrive le caratteristiche di **un'istanza dell'informazione da rappresentare**
- I valori contenuti nell **colonne** descrivono sempre la **stessa proprietà delle istanze** di informazione da rappresentare

**Definizione 5.1** (Definizione di relazione come insieme di ennulple (Lists))**.** Dati $n$ insiemi di valori (domini) $D_1, \ldots, D_n$, con $n > 0$ e indicato con $D_1 \times \ldots \times D_n$ il loro prodotto cartesiano:

$$D_1 \times \ldots \times D_n = \{(v_1, \ldots, v_n) \mid v_1 \in D_1 \wedge \ldots \wedge v_n \in D_n\}$$

una relazione $\rho$ di grado $n$ è un qualsiasi **sottoinsieme di** $D_1 \times \ldots \times D_n$:

$$\rho \subseteq D_1 \times \ldots \times D_n$$

dove $(v_1, \ldots, v_n)$ è una ennupla della relazione e $|\rho|$ è la cardinalità della relazione (numero di ennuple).

**Nota**:

- I domini $D_1, \ldots, D_n$ possono essere a **cardinalità infinita**, mentre le relazioni sono sempre a **cardinalità finita**
- Dalla definizione si deduce inoltre che:
  - **Non è definito alcun ordinamento** sulle ennuple di una relazione
  - **Non sono ammessi duplicati** di una ennupla
  - Nella definizione di relazione come insieme di ennuple, i valori delle ennuple sono ordinati

**Esempio 5.2.** Riprendendo l'esempio 5.1, la relazione può essere vista come insieme di ennuple (relazione città):

$$\rho = \left\{ \begin{array}{l} (\text{Milano}, 20100, 1300000)\,, \\ (\text{Verona}, 37100, 350000)\,, \\ (\text{Brescia}, 25100, 250000) \end{array} \right\}$$

dove:

$$\rho \subseteq D_1 \times D_2 \times D_3$$

### 5.2.1 Accesso ai valori di una ennupla

Se $t$ è una ennupla $(v_1, \ldots, v_n)$ il valore posto in i-esima posizione si indica con la notazione:

$$t[i]$$

Questa modalità di accesso ai valori però non è efficace per l'uso pratico delle relazioni. Si preferisce quindi assegnare un **nome alle colonne**; ciò conduce all'introduzione della definizione di relazione come insieme di **tuple**.

**Definizione 5.2** (Definizione di relazione come insieme di tuple (Mappings))**.** Sia $X$ un insieme di nomi e sia $\Delta$ l'insieme di tutti i domini di base ammessi dal modello. Si definisce la funzione:

$$DOM : X \to \Delta$$

Tale funzione associa ad ogni nome $A \in X$ un dominio $DOM(A) \in \Delta$. I nomi di $X$ si definiscono **attributi**. Una tupla $t$ su $X$ è una funzione:

$$t : X \to \bigcup_{A \in X} DOM(A)$$

dove:

$$t[A] = v \in DOM(A)$$

Quindi una relazione su X è un **insieme di tuple su** $X$, dove X è l'insieme di attributi della relazione.

**Nota**:

- Una relazione è un **insieme** di tuple e quindi **non può contenere tuple duplicate**
- I domini per gli attributi possono essere solo domini di base, **non sono ammessi altri domini**, nè **il prodotto cartesiano di domini**. Essenzialmente non sono ammessi attributi annidati
- In generale una base di dati relazionale è composta da **più relazioni**

**Esempio 5.3.** Consideriamo l'esempio 5.1. Definiamo l'insieme di attributi:

$$X = \{\text{Nome}, \text{CAP}, \text{Abitanti}\}$$

Definiamo il dominio di ciascun attributo:

$$\begin{aligned} DOM(\text{Nome}) &= \text{Stringhe di caratteri} \\ DOM(\text{CAP}) &= \text{Numeri interi} \\ DOM(\text{Abitanti}) &= \text{Numeri interi} \end{aligned}$$

La tabella della relazione è rappresentata come un insieme di tuple:

$$\rho_X = \{t_1, t_2, t_3\}$$

dove:

$$\begin{array}{lll} t_1[\text{Nome}] = \text{Milano} & t_1[\text{CAP}] = 20100 & t_1[\text{Abitanti}] = 1300000 \\ t_2[\text{Nome}] = \text{Verona} & t_2[\text{CAP}] = 37100 & t_2[\text{Abitanti}] = 350000 \\ t_3[\text{Nome}] = \text{Brescia} & t_3[\text{CAP}] = 25100 & t_3[\text{Abitanti}] = 250000 \end{array}$$