La progettazione delle basi di dati

Prof. Fedeli Massimo

La progettazione del database

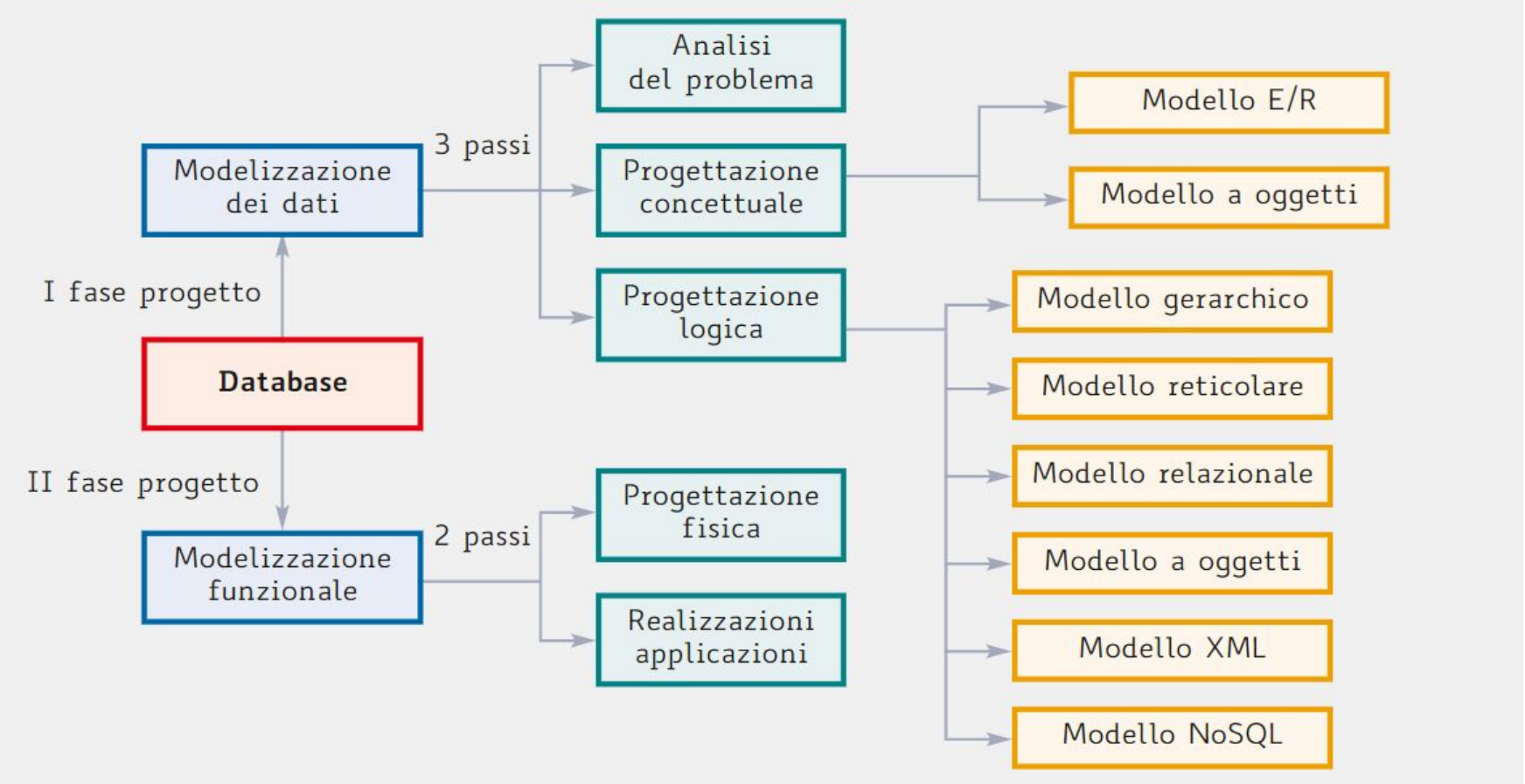
I passi principali per progettare un database si possono così schematizzare:

1. **analisi del problema**;
2. **progettazione concettuale** del database → **modello E-R**;
3. **progettazione logica** del database → **schema logico**;
4. **progettazione fisica** e implementazione;
5. **realizzazione delle applicazioni** → (Java, VB,PHP, etc)

Ognuno di questi passi presenta delle **criticità** e **implica** un insieme di operazioni anche complesse tra cui:

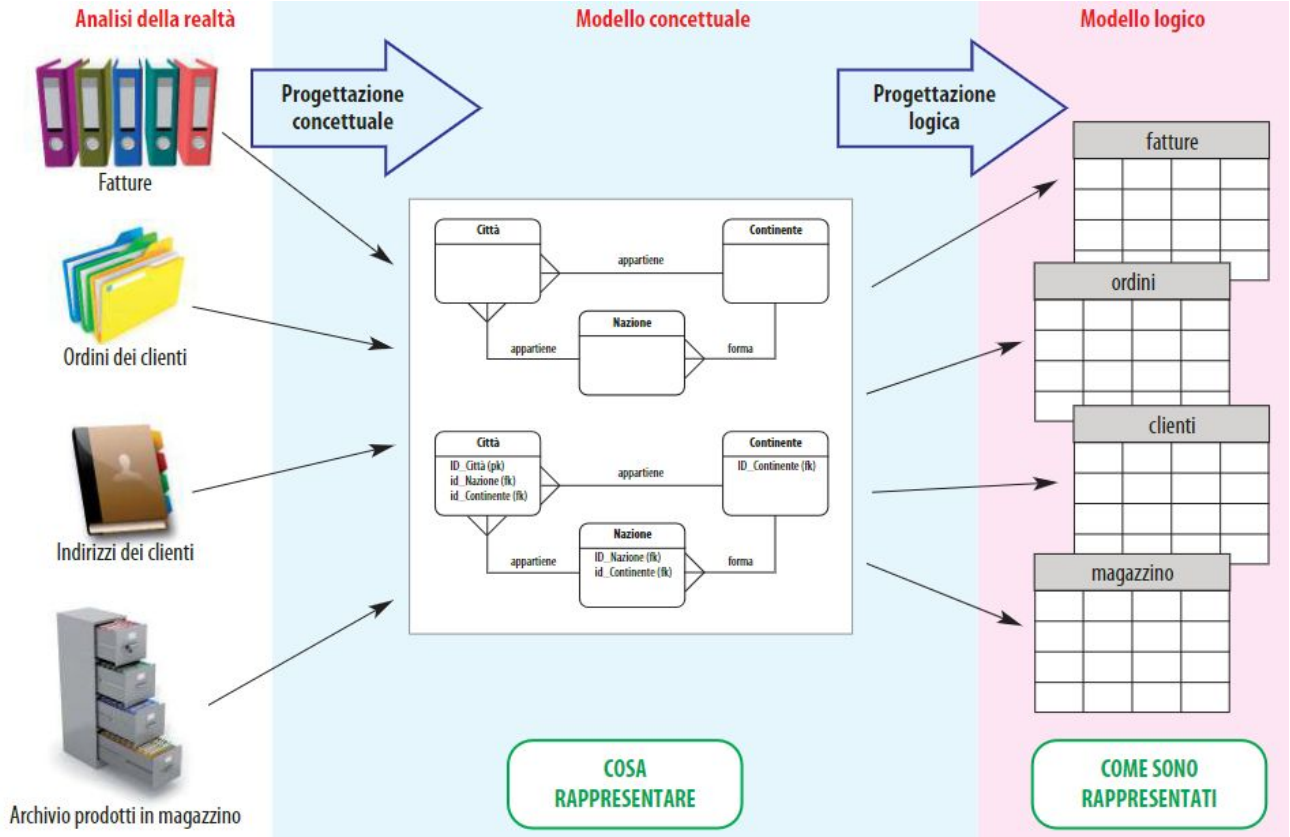
- la **fase di sviluppo** del database
- la **fase di sviluppo** dell'applicazione
- i **test** di funzionamento di entrambi, il collaudo ecc

Le fasi della progettazione di un database



Le fasi della progettazione di un database

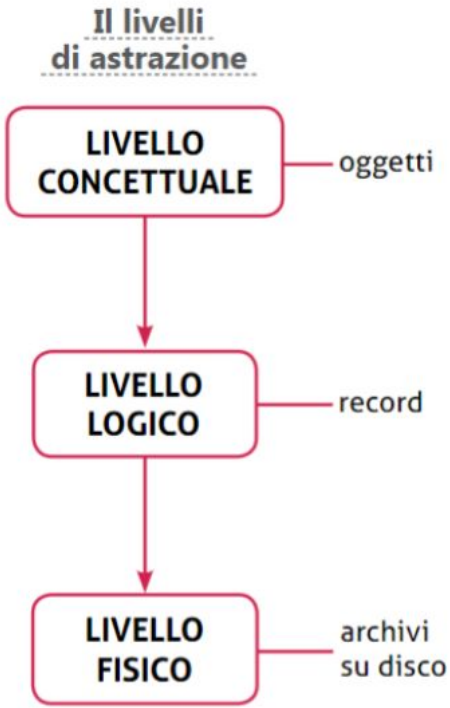
L'output della progettazione concettuale è il **modello concettuale.**



I livelli di astrazione del modello di database

La **progettazione di un modello** di dati avviene a diversi livelli di astrazione dal problema considerato:

- il **livello concettuale** (o livello di oggetti) rappresenta la **realtà dei dati** e le **relazioni** tra essi attraverso uno schema;
- il **livello logico** (o livello di record) rappresenta il modo attraverso il quale i dati sono organizzati negli archivi: descrive quindi la composizione e il **formato dei dati** secondo la loro struttura logica.
- Il **livello fisico** rappresenta l'**effettiva installazione degli archivi su disco**: esso indica l'ubicazione dei dati nelle memorie di massa.



Il **livello fisico** è quindi l'implementazione del livello logico sui supporti per la registrazione fisica dei dati e si occupa di oggetti quali partizioni, puntatori, blocchi fisici, cluster, indici.

Analisi preliminare alla modellazione

L'**analisi preliminare** per la **modellazione** dei dati avviene solitamente cercando di individuare le **esigenze del cliente** o il **dominio dell'applicazione**, cioè quali informazioni devono essere salvate e in che modo queste informazioni verranno manipolate dall'utente.

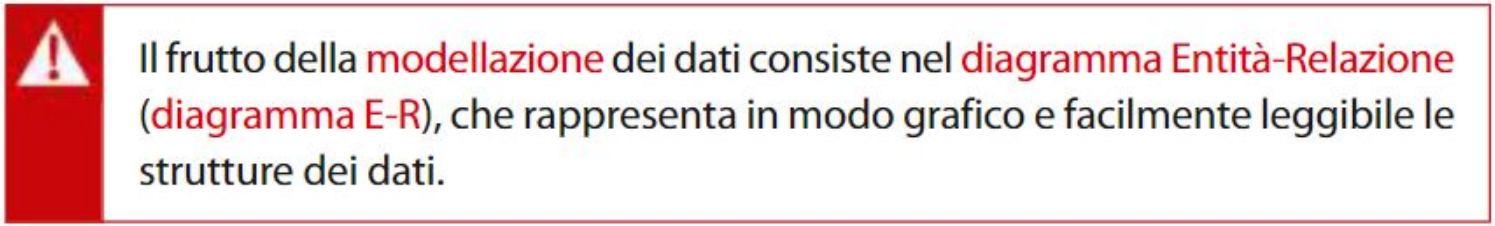
Le tecniche di **analisi del problema** sono da considerarsi valide e applicabili anche alla progettazione dei database: naturalmente bisognerà prestare maggiore attenzione al dato e quindi spesso la tecnica **bottom-up** è da preferirsi a quella **top-down**

La fase di modellazione

Al termine dell'analisi inizia la prima fase di modellazione, che è quella **concettuale**; per attuarla, si può far ricorso ai due seguenti modelli:

- modello **entità-relazione;**
- modello a **oggetti.**

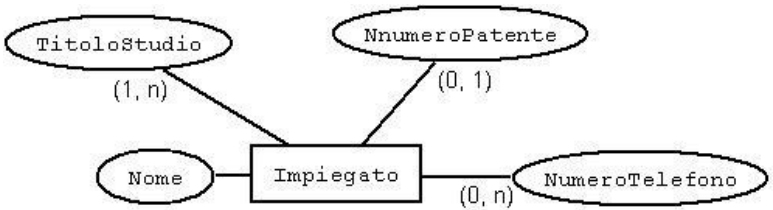
Benché il secondo modello stia cominciando a rivelarsi interessante da un punto di vista commerciale e non solamente accademico, la stragrande maggioranza delle applicazioni esistenti ricorre all'approccio **Entità-Relazione(E-R)**.



Modello entità-relazione

Il modello **Entità-Relazione** consente di superare questo ostacolo in quanto è assolutamente **indipendente dal linguaggio scritto o parlato** e permette quindi a tutti di comprendere la struttura del database.

Di norma, al modello **Entità-Relazione** viene affiancato un **documento tecnico** che descrive in maggior dettaglio, usando un linguaggio naturale, i concetti espressi graficamente dal modello Entità-Relazione.



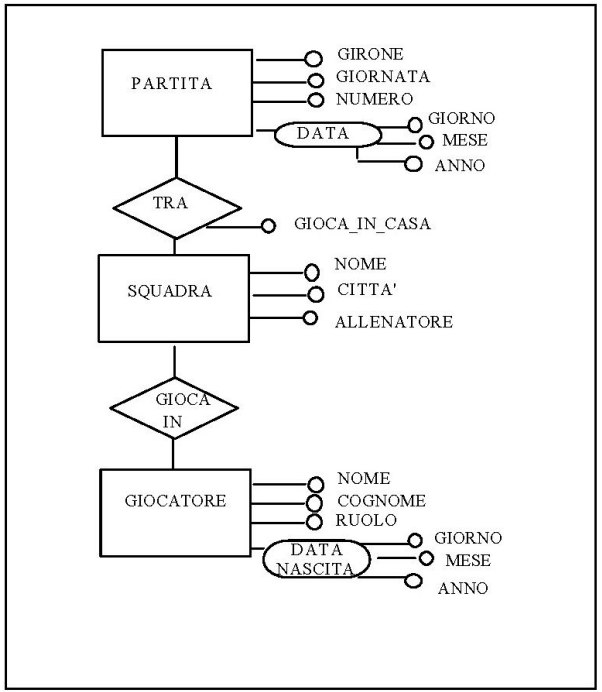
Progettazione concettuale →modello entità-relazione

- Il **primo scopo** del modello **Entità-Relazione** è quello di fornire la **rappresentazione grafica** di

*tutti gli oggetti che fanno parte di un **database***

così che il **flusso delle informazioni** possa essere seguito e verificato prima di sviluppare l'applicazione.

- In **secondo luogo**, questo modello può essere usato dagli sviluppatori per creare il **database fisico** e tutti gli oggetti che ne fanno parte.



Progettazione concettuale →modello entità-relazione

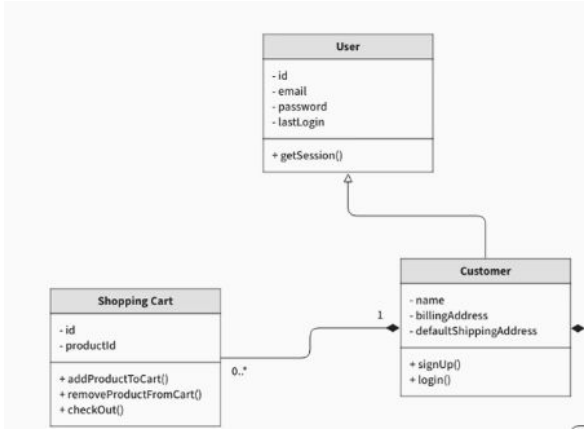
- **Correttezza** (Uso corretto degli strumenti): utilizzo appropriato degli strumenti di modellazione concettuale secondo le loro finalità.
- **Completezza** (Modellazione di tutti gli aspetti rilevanti della realtà): rappresentazione di tutti gli aspetti significativi del dominio applicativo da modellare.
- **Chiarezza** (Modello leggibile e informazioni comprensibili): facilità di lettura del modello e comprensione delle informazioni rappresentate.
- **Indipendenza** (Non dipendente dagli strumenti informatici successivi)

Il modello concettuale non deve dipendere da implementazioni o strumenti informatici specifici.

Correttezza

Il modello concettuale deve utilizzare gli **strumenti di modellazione** (come diagrammi ER, UML, BPMN, ecc.) secondo la loro finalità specifica, senza forzature o interpretazioni errate.

Non si tratta solo di "usare lo strumento": un **diagramma ER**, ad esempio, serve a rappresentare relazioni tra entità, non flussi di processo. Usarlo per descrivere un workflow sarebbe un errore concettuale. **Coerenza semantica**: Ogni simbolo (es. rombo per le relazioni, rettangoli per le entità) deve rispettare le convenzioni stabilite.



Indipendenza

I modello concettuale deve essere astratto e neutrale, non vincolato a tecnologie, DBMS o linguaggi di programmazione specifici. Si deve descrivere cosa fare, non come implementarlo.

Vantaggi:

Longevità (il modello sopravvive ai cambi tecnologici).

Flessibilità (può essere implementato in diversi modi).

Esempio pratico:

Dipendente: Scrivere nel modello "Usare PostgreSQL con trigger per gestire cancellazioni".

Indipendente: Definire una regola di business come "Una prenotazione cancellata deve aggiornare la disponibilità della risorsa", senza menzionare tecnologie.

Fasi della progettazione concettuale

Caratteristiche della progettazione concettuale

- **Rigorosa** → per non lasciare dubbi in merito alle caratteristiche della base di dati che si sta progettando;

- Espressa con **formalismi semplici** → per consentire la lettura e la comprensione anche da parte di utenti **non tecnici.**

Gli **utenti** devono infatti essere certi che i **progettisti** abbiano compreso a fondo tutte le loro esigenze.

Il modello **entità/associazioni(relazioni)** ha queste caratteristiche e si concretizza in un documento con **schemi grafici**.

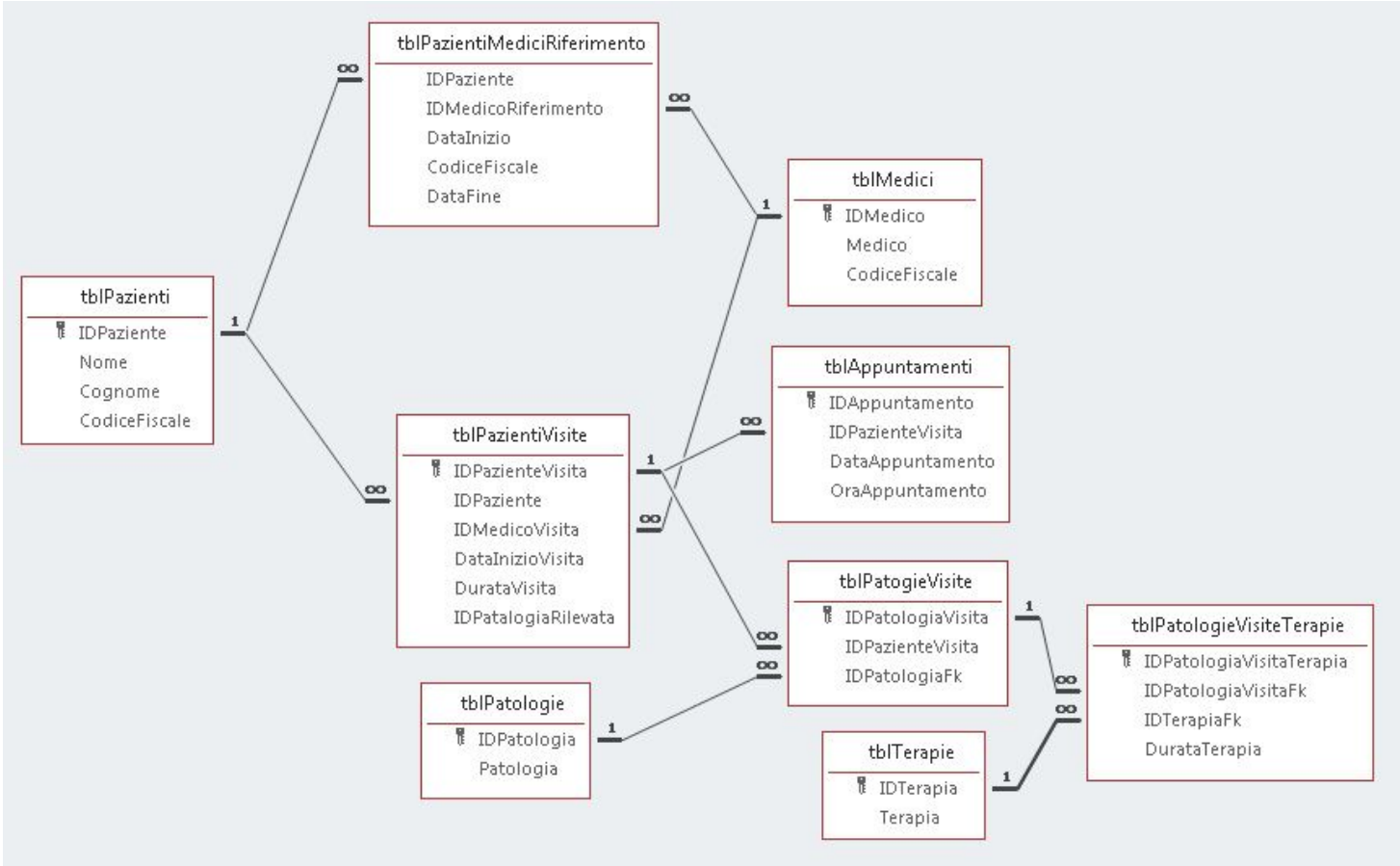
La progettazione concettuale → il modello concettuale

- Un aspetto rilevante del **modello concettuale** sono **concetti** e **formalismi** utilizzati nella costruzione del modello **entità/associazioni**.
- Il modello concettuale, pur essendo molto utilizzato nella **progettazione concettuale**, **non ha una rappresentazione standardizzata**.
- Il modello **entità/associazioni** è composto da **entità**, **associazioni** e **attributi** e che cosa si intenda con tali termini: esistono però diversi modi di rappresentarli.
- La notazione classica e la notazione standard **UML**.

Chi realizza il modello concettuale? Il database designer

Il **database designer** è responsabile dell'**astrazione dei dati** dal **mondo reale** a partire dall'**analisi dei requisiti** fino a ottenere la corretta modellazione degli stessi nello **schema concettuale** e successivamente nello **schema logico**.

Schema logico



I compiti del database designer

- Il primo compito di un database designer consiste nell'analizzare le informazioni raccolte durante l'analisi dei requisiti.
- Il suo principale obiettivo è costruire il modello di base, che andrà poi **raffinato** e **ristrutturato** fino al suo completamento.

Le prime operazioni che il modellatore esegue hanno il compito di classificare gli oggetti come entità oppure attributi

Si procede partendo dalla documentazione del progetto

- raccolta e analisi della documentazione
- definizione del glossario dei termini

Analisi della documentazione

Possiamo catalogare la documentazione utilizzata per la **definizione dei requisiti**:

- **documentazione specifica prodotta per il progetto**
  - le note delle riunioni tecniche e le richieste del cliente
  - gli appunti sulle interviste agli utenti finali
  - la documentazione scritta predisposta appositamente
- **documentazione esistente**
  - le normative generali e del settore
  - i regolamenti interni
  - le procedure aziendali
- **sistema esistente**
  - il sistema da rimpiazzare
  - le specifiche di integrazione con sistemi esistenti

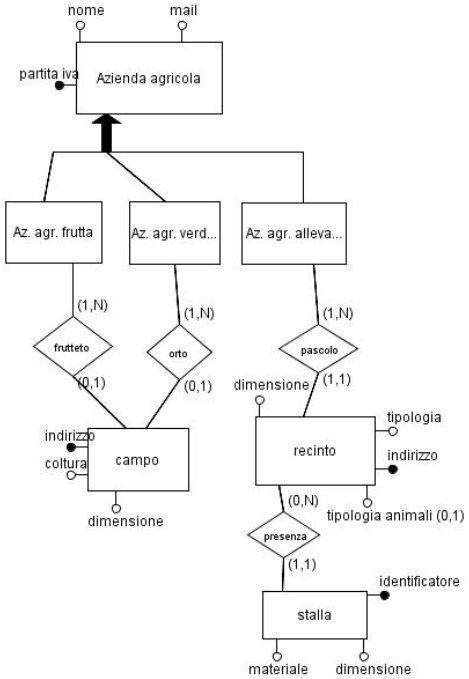
Identificare le entità

Identificare le entità

- Le entità sono i primi elementi da determinare nella progettazione concettuale di un database

- Viene considerata un'entità:
  – qualunque oggetto per il quale è necessario salvare alcuni attributi;
  – qualsiasi oggetto che deve essere rappresentato in un database.

- In particolare, sono entità:
  – qualsiasi persona, luogo, cosa, evento o concetto distinguibile, sul conto del quale le informazioni vanno memorizzate;
  – qualsiasi cosa per la quale salviamo informazioni (per esempio fornitori, macchine, impiegati, numero posti in aereo ecc.).



Identificare gli attributi

Gli attributi descrivono un'entità: corrispondono ai **campi** dei record.

Regole per individuare gli attributi:

1. Gli attributi devono essere atomici
2. Gli attributi derivati non dovrebbero essere memorizzati.
3. Utilizzare codici per classificare gli attributi ogni volta che si presenta tale opportunità.

Identificare gli attributi

Entità: Studente

Attributi:

- NumeroMatricola (numero intero)
- Nome (stringa)
- Cognome (stringa)
- DataNascita (data)
- CorsoDiLaurea (stringa)
- MediaVoti (numero decimale)
- Indirizzo (stringa)
- NumeroTelefono (stringa)
- Email (stringa)

Scegliere correttamente i nomi

- I **nomi** degli oggetti devono essere **unici**;
- Devono avere un **significato** per l'utente finale;
- Devono contenere il numero minimo di parole di cui si ha bisogno per descrivere univocamente e accuratamente l'oggetto.
- Per le **entità** e gli **attributi** i nomi sono usati generalmente al **singolare**
- Le **associazioni**/**relazioni** sono tipicamente descritte attraverso verbi.

Esempio di entità

Una semplice relazione tra Città e Persone può essere così rappresentata in UML.

oppure con un unico termine per la relazione nel diagramma E-R.

Identificare le associazioni

Documentare il progetto: matrici tra entità e attributi

Si vogliono definire le associazioni e gli attributi per organizzare i dati relativi ai dipendenti di una società allo scopo di gestire le informazioni concernenti i progetti seguiti dagli impiegati, le attività da svolgere all'interno di un progetto e le conoscenze necessarie per i progetti.

Matrice entità-entità

	IMPIEGATO	PROGETTO	ATTIVITÀ	CONOSCENZE
Impiegato		X		
Progetto	X		X	X
Attività		X		
Conoscenze	X			

Matrice entità-attributo

	IMPIEGATO	PROGETTO	ATTIVITÀ	CONOSCENZE
Matricola impiegato	X			
Nome impiegato	X			
Mansione				
Codice progetto		X		
Nome progetto		X	X	
Attività			X	
Descrizione attività			X	
Conoscenza				X
Specializzazione				X

Individuare le relazioni

A partire dalla matrice entità-entità si deve:

– definire e individuare tutte le associazioni;

– classificare le associazioni in termini di:

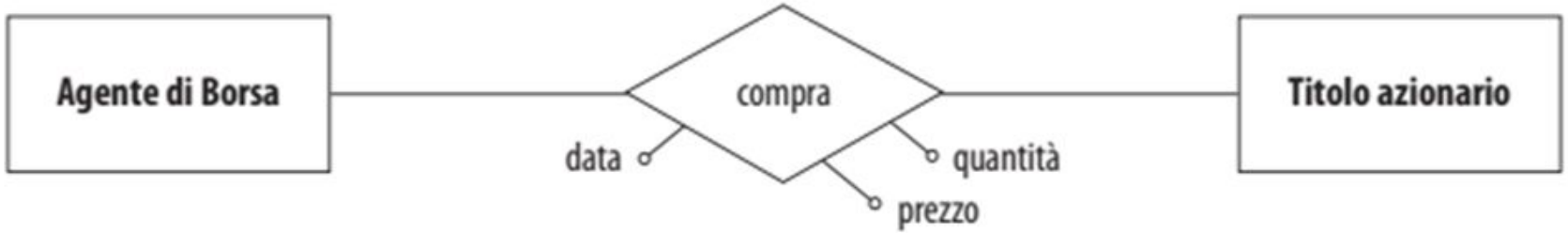
- cardinalità, per quantificare le relazioni tra le entità misurando come le istanze sono relazionate tra loro
- opzionalità, per verificare se la relazione deve è obbligatoria o no
- direzione, per individuare la direzione della relazione
- dipendenza, per stabilire se una relazione dipende da una o più altre relazioni.

Individuare le relazioni

- Una volta individuate le entità, si definiscono le possibili gerarchie in termini di:
  - generalizzazione, se un caso è più generale di un altro
  - specializzazione, se un caso è più particolare di un altro.

- La rappresentazione grafica dello schema ci permette di poter aggiungere gli eventuali attributi nel caso che questi siano legati specificamente alla relazione e non alle **entità**

Individuare le relazioni



Documentare il progetto: matrici tra entità e attributi

Per individuare le relazioni si analizzano i verbi riportati nelle frasi scritte in fase di analisi:

- *gli studenti ricevono voti*
- *una persona possiede uno o più* ***telefoni***

Le **associazioni** stabiliscono legami tra entità.

Rappresentazione delle entità

Le entità

- L'**entità** è un oggetto (concreto o astratto) che ha un significato anche quando viene considerato in modo isolato ed è di interesse per la realtà che si vuole modellare.
- Sono entità una **persona**, un **modello di smartphone**, un **movimento contabile**, una prova sostenuta da uno studente o una studentessa
- Le entità possono essere classificate secondo un **criterio di omogeneità** definendo il tipo di entità attraverso un nome.

Le entità

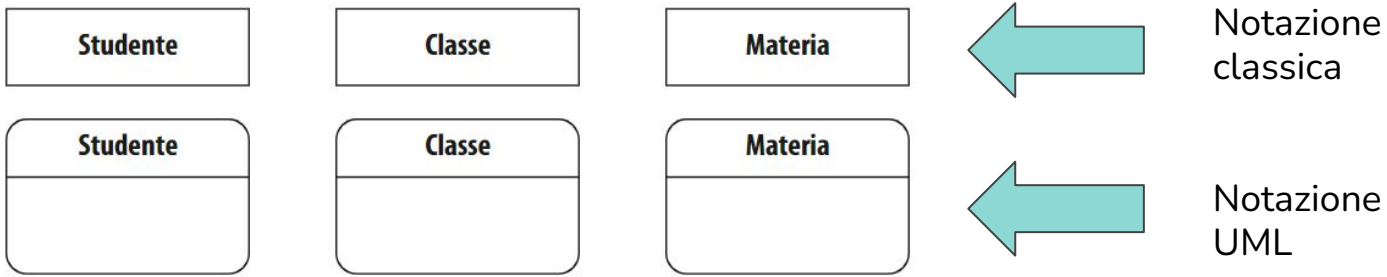
- Gli **studenti** e le **studentesse** di una scuola sono classificabili nel tipo di **entità** Studente, i diversi modelli di smartphone sono classificabili nel tipo di entità Smartphone.
- Singoli studenti, singole studentesse o singoli smartphone rappresentano un'istanza del tipo di entità Studente o Smartphone: l'**istanza** non è altro che un esemplare di un tipo di entità.

Le entità

- Un'**entità** serve per raccogliere informazioni e quindi possiamo pensarla come un oggetto che rappresenta un "**gruppo omogeneo di informazioni**".
- A ciascuna entità diamo un **nome** che ci permette di identificare ogni **istanza** di quella entità: per esempio, gli studenti di una scuola sono classificabili nell'entità **Studente**, le materie studiate nell'entità **Materia** e la classe di appartenenza nell'entità Classe e così via

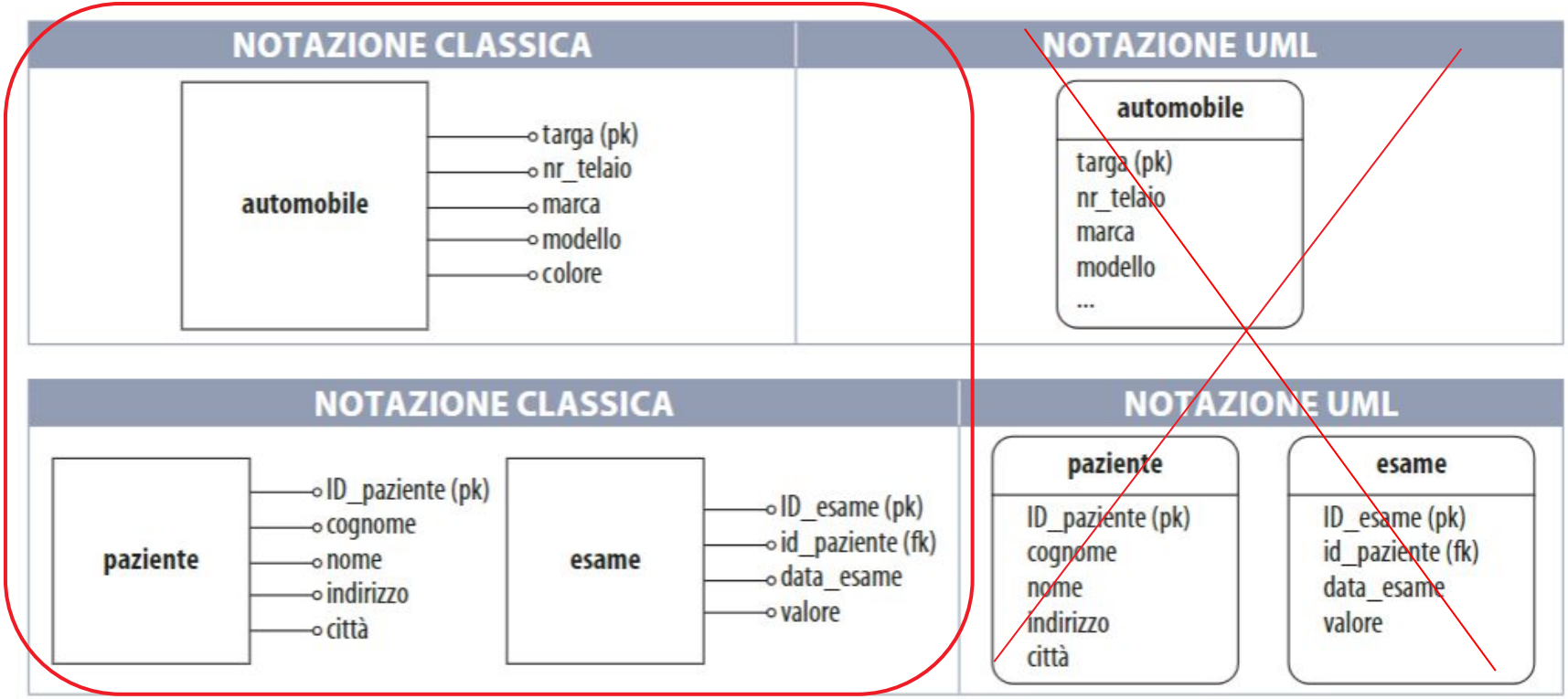
Le entità sono "le candidate" a diventare le tabelle del database relazionale.

Le tre entità appena elencate – Studente / Classe / Materia – vengono rappresentate graficamente in questo modo:



Le entità - confronto tra notazione classica e UML

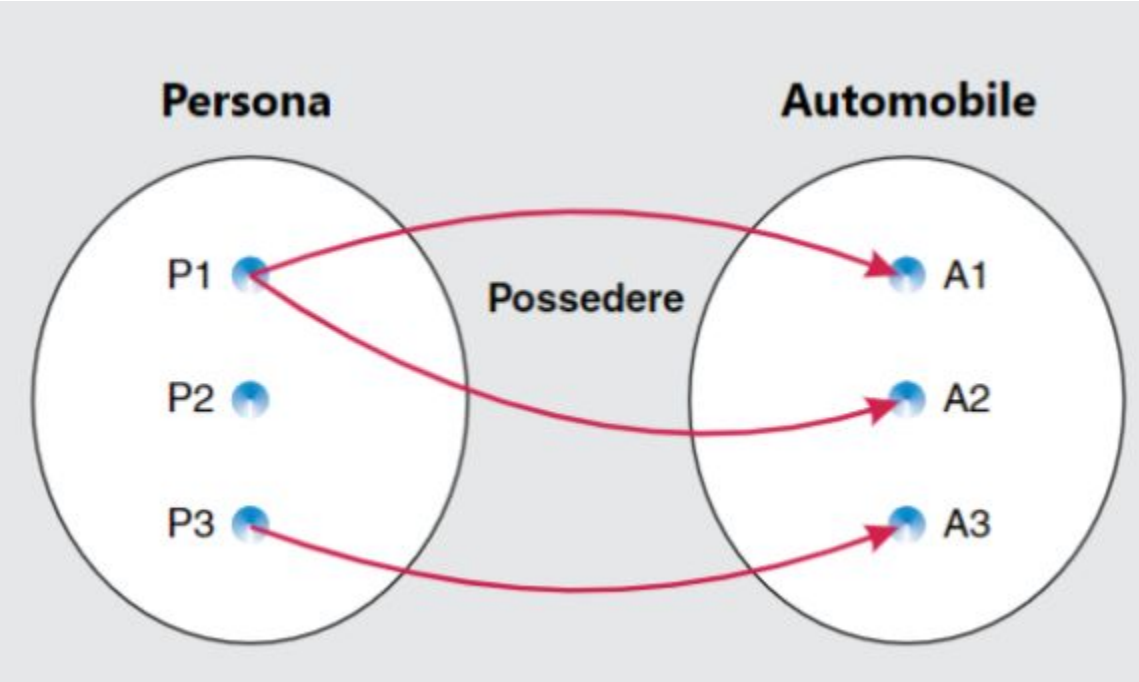
A volte viene preferito aggiungere all'attributo le iniziali pk per le chiavi primarie e fk per le chiavi esterne.



Rappresentazione delle relazioni

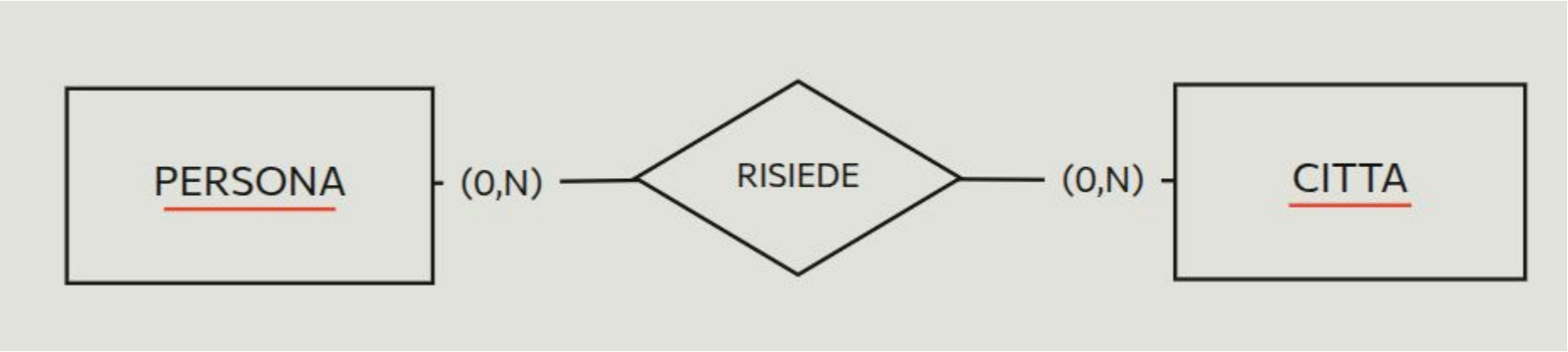
Le associazioni (o relazioni)

- Una associazione è un legame che stabilisce un'interazione tra **entità**.



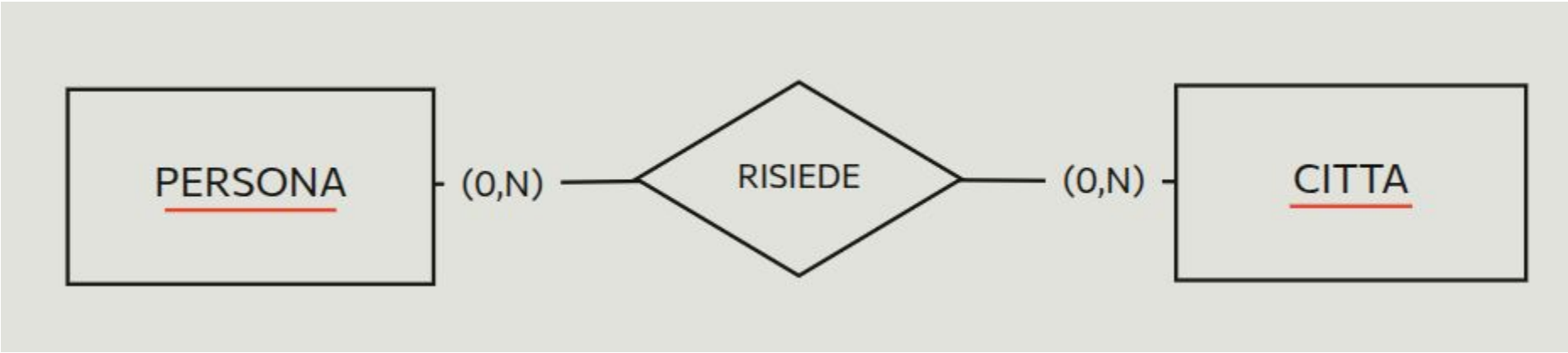
Le associazioni

- La rappresentazione grafica convenzionalmente usata per indicare un’associazione un **rombo** che collega le due entità.
- il **nome dell’associazione** compare all’interno del rombo.



Le associazioni(relazioni)

- Di norma i **nomi delle entità** sono **sostantivi** mentre i nomi delle **associazioni** sono **verbi**: in questo modo si cerca di stabilire una corrispondenza tra la rappresentazione delle associazioni e le frasi del linguaggio naturale che le descrivono.



Caratteristiche delle associazioni tra entità

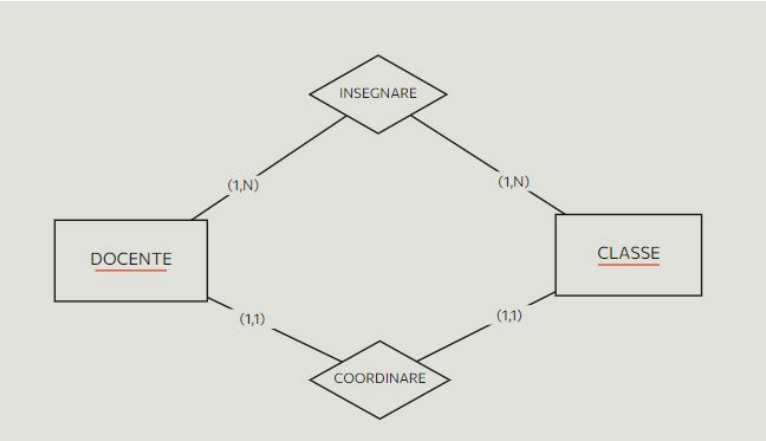
Grado delle associazioni

- Le associazioni hanno un **grado**.
- Il grado è dato dal **numero di entità** che partecipano all'associazione.
- L'associazione tra **Automobile** e **Persona** è un'associazione di **grado 2**, detta **associazione binaria**,ma naturalmente esistono associazioni di grado 1, 3 e di grado superiore.

Le associazioni binarie sono le più diffuse.

I ruoli delle relazioni multiple tra stesse associazioni

- **Due entità** possono essere collegate da **più di una relazione**, come nel caso delle entità **Docente** e **Classe** tra le quali esistono le due associazioni **Insegnare** e **Coordinare**.
- In questo caso nella rappresentazione si possono evidenziare, **mediante un nome**, i **ruoli** che le **entità** giocano nelle due relazioni.



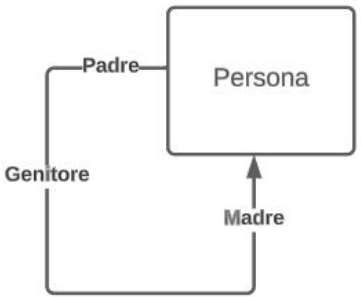
Il ruolo nelle associazioni ricorsive

- Le associazioni di **grado 1** sono associazioni tra un'entità e se stessa.

Si consideri per esempio l'entità **Persona** e l'associazione di **maternità** (o paternità) che collega una **madre** con i **propri figl**i (o un padre con i propri figli).

In queste associazioni l'entità **Persona** partecipa con **ruoli diversi** all'associazione.

Le associazioni di questo tipo si dicono **associazioni ricorsive.**

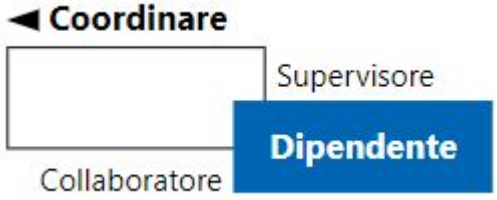


Le associazioni ricorsive

- Altro esempio di associazione ricorsiva nella quale l'entità **Dipendente** partecipa all'associazione **Coordinare** nel duplice **ruolo** di **Supervisore** e di **Collaboratore**.

Utilizzando i nomi dei diversi ruoli che un dipendente assume nell'associazione Coordinare, la realtà descritta in figura può essere espressa in **linguaggio naturale** con le frasi:

- un **supervisore** **può coordinare** uno o più collaboratori;
- un **collaboratore** **è coordinato** da un supervisore



Le relazioni/associazioni binarie

Le relazioni binarie, ovvero le associazioni tra due entità, sono le relazioni più comuni nel mondo reale e, di conseguenza, saranno quelle che verranno utilizzate nei nostri progetti.

Supponiamo di avere le entità **studente** e **città**, che raccolgono le informazioni sugli studenti e sulla città dove abitano.
Un esempio di relazione binaria è la residenza, che associa ogni **studente** alla **città** in cui risiede.

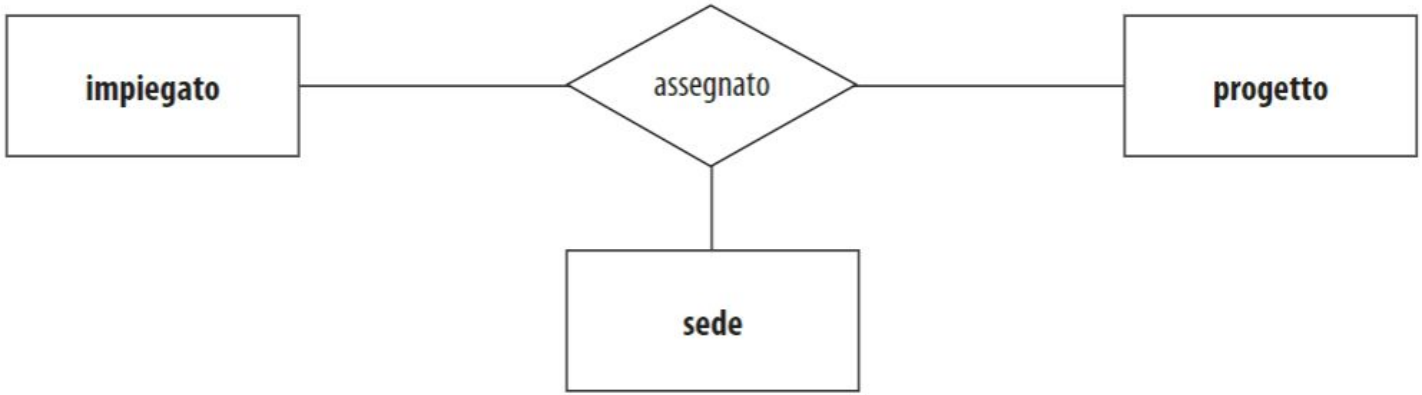
NOTAZIONE CLASSICA

Le associazioni ternarie

Una relazione ternaria implica tre entità e viene usata quando quella binaria è inadeguata: molti approcci di modellazione *riconoscono solo relazioni binarie* e in essi le relazioni ternarie o n-arie vengono decomposte in due o più relazioni binarie.

ESEMPIO

NOTAZIONE CLASSICA



Attributi

Gli attributi

Le principali **caratteristiche di un attributo** di un'entità sono:

- il **formato**,che indica il tipo di valori che assume;
  - i tre **formati di base** sono: **carattere**, **numerico**, **data/ora**;
- la **dimensione**, che indica la quantità massima di caratteri o cifre inseribili quando si valorizza l'attributo;
- l'**opzionalità**: l'attributo è **obbligatorio** se deve avere valore non nullo, come il nome di una persona in un'anagrafica, oppure **facoltativo** se sono accettabili valori nulli, come il titolo di studio.

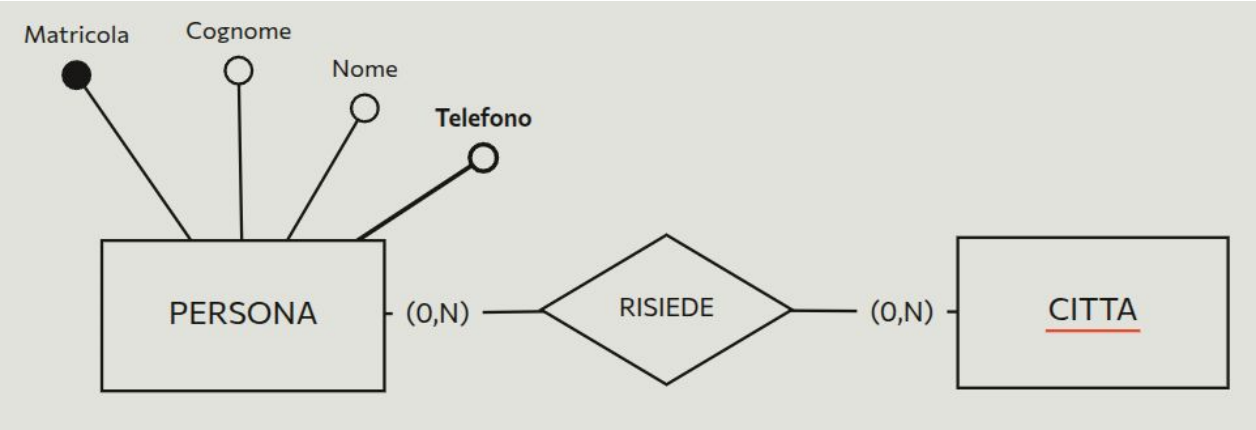
Le proprietà delle entità e delle associazioni sono descritte attraverso gli **attributi**.

ESEMPIO

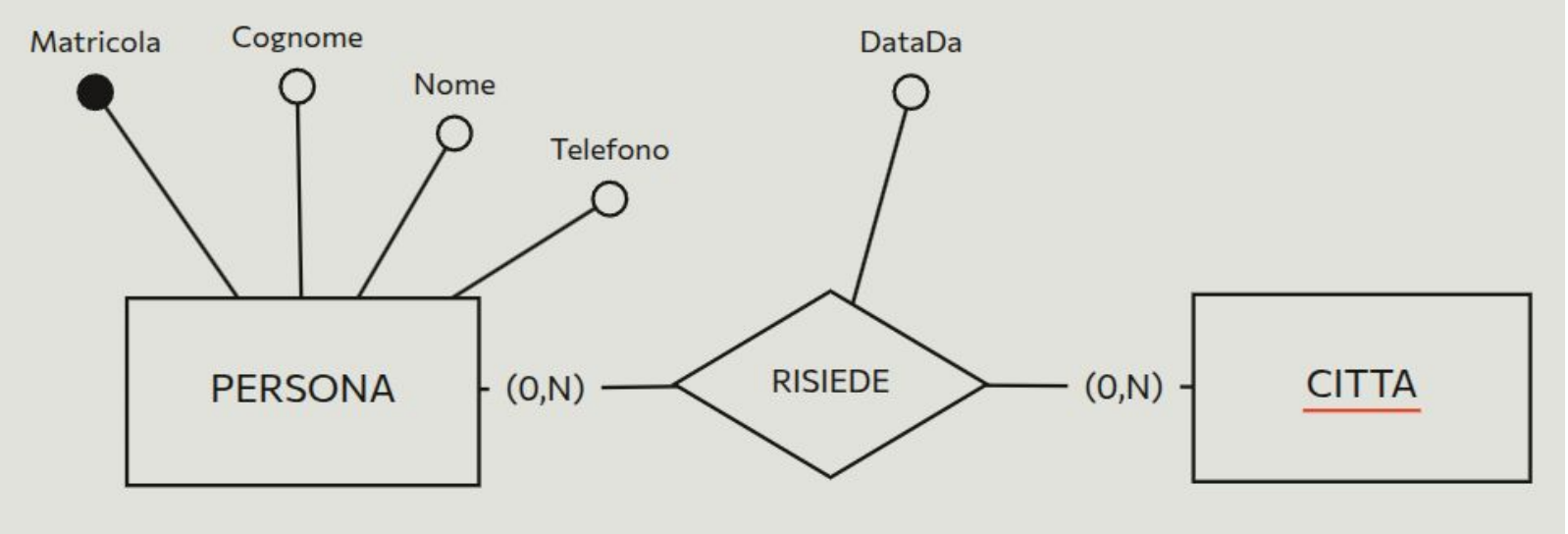
Attributi per l'entità *Automobile* sono: *Modello, Produttore, Cilindrata, Potenza, PrezzoListino.*

Il dominio di un attributo

- L'**insieme dei possibili** valori assunti da un attributo si chiama **dominio** dell'attributo. I valori appartenenti al dominio sono omogenei tra loro, cioè sono dello stesso tipo.
- Gli **attributi** sono indicati graficamente con dei fiammiferi stilizzati.



Gli attributi delle relazioni



- Anche le associazioni(relazioni) possono avere **attributi**.

Attributi delle associazioni - esempio

- Si consideri la situazione di una **concessionaria di automobili** che vende alle persone (**clienti**) i diversi modelli di automobili (**prodotti**).

- L'**associazione** di nome **Acquistare** associa una

  - **persona** ← → **modello di automobile** acquistato.

- L'acquisto avviene in una certa data (**DataAcquisto**), a un certo prezzo (**PrezzoAcquisto**) e con un certo numero di targa per l'immatricolazione (**Targa**).

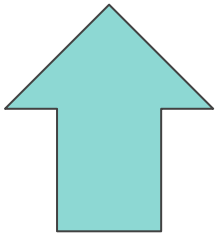
- **PrezzoAcquisto** **non è** un attributo di **Automobile** (un'automobile è caratterizzata da un **prezzo di listino**, ma potrebbe avere un prezzo di acquisto differente per ogni vettura venduta).

Attributi delle associazioni - esempio

- Analogamente **PrezzoAcquisto** **non è** un attributo della persona.

- Un ragionamento analogo si può fare per gli attributi **DataAcquisto** e **Targa**.

Essi sono caratteristici dell'abbinamento tra una persona e l'automobile acquistata, cioè della relazione **Acquistare** tra le entità **Persona** e **Automobile**.



ATTRIBUTI DELL' ASSOCIAZIONE

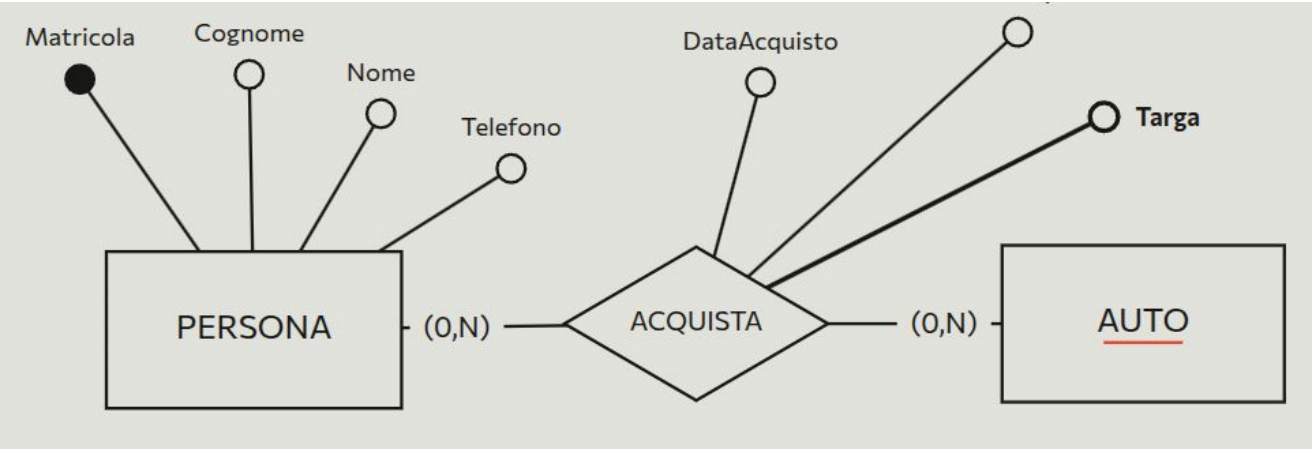
Attributi delle associazioni - esempio

- La presenza di un'**associazione con attributi** potrebbe essere indicativa della presenza di **un'altra entità** oltre a quelle identificate.

Un progettista di dati potrebbe modellare la precedente situazione per mezzo delle entità **Persona**, **Automobile**, **Acquisto** e di due **associazioni** che collegano le tre **entità**:

- una tra **Persona** e **Acquisto**
- una tra **Acquisto** e **Automobile**

Attributi delle associazioni - esempio



- La differente visione del problema non porta a una **rappresentazione corretta** in un caso ed **errata** nell'altro.
- Non solo: applicando le regole di passaggio dal modello concettuale al modello logico, le due rappresentazioni portano alle stesse strutture di dati.

Attributi delle associazioni

- Una **regola molto importante** richiede di definire nella fase iniziale **attributi elementari** e quindi di non definire gli attributi che si ottengono con le elaborazioni, cioè gli **attributi derivati**.
- Il mancato rispetto della regola provoca **inefficienza** dovuta alle elaborazioni necessarie per aggiornare questo tipo di attributi. Es. Età della persona

ESEMPIO

L'età di una persona è un attributo derivato dall'attributo elementare della data di nascita; il saldo di un conto corrente è derivato dalla somma algebrica degli importi dei movimenti effettuati sul conto.

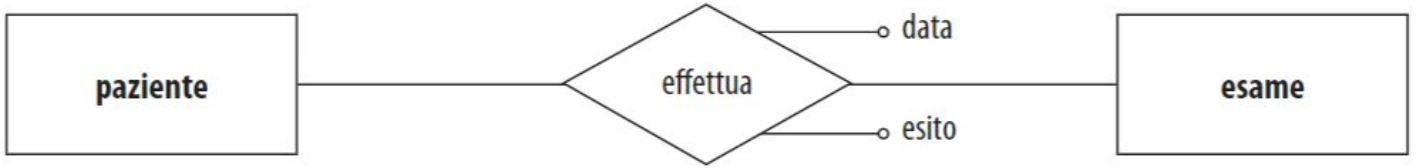
Relazioni con attributi confronto notazione classica e UML

Esaminiamo un primo caso, dove individuiamo l'**attributo** della relazione tra **studente** e **materia**, cioè "lo studente viene interrogato e prende un voto in una materia";

NOTAZIONE CLASSICA

Prendiamo ora in considerazione un altro esempio, e individuiamo l'attributo della relazione tra **paziente** ed **esame**, cioè "*il paziente* ***oggi*** *effettua un esame con un certo esito*".

NOTAZIONE CLASSICA



Chiave primaria

La chiave primaria

Chiave (Key) / Chiave primaria (Primary Key):
Un attributo o un insieme minimo di attributi che identifica in modo univoco un'istanza di un'entità.

ESEMPI:

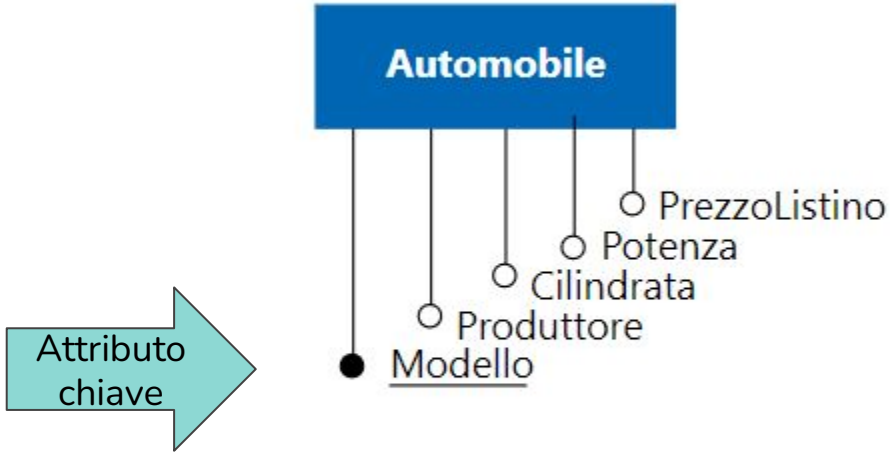
Codice prodotto
Numero di matricola dipendente
Codice studente + Data + Codice materia
Per l'entità Persona, la chiave primaria è il Codice Fiscale.
Per l'entità Automobile, la chiave primaria è il Modello.

La chiave primaria

- La **chiave primaria** di un'entità viene riconosciuta dalla presenza dell'acronimo: **{PK} (Primary Key)**, posto tra parentesi graffe accanto all'attributo chiave.
- Nel caso di chiave formata da più attributi, l'acronimo **{PPK} (Partial Primary Key)** è posto accanto a ognuno degli attributi che compongono la chiave.

La diverse rappresentazioni degli attributi

Nelle altre rappresentazioni il simbolo grafico convenzionalmente usato per rappresentare l'**attributo** è la linea che parte dall'entità e termina con il nome e un piccolo cerchio.



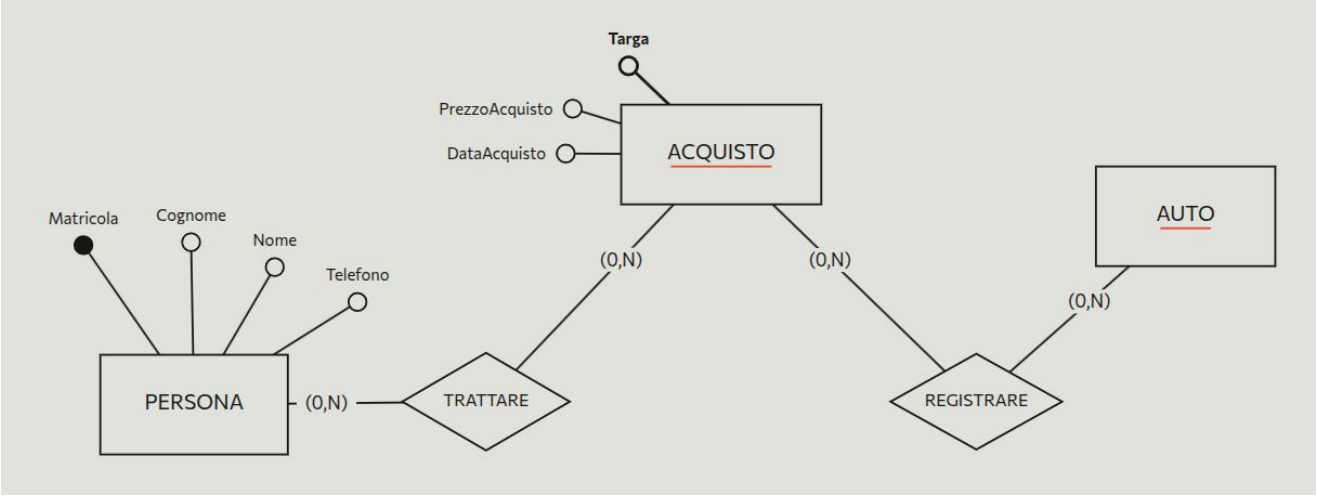
NOTA BENE

Nella descrizione grafica, gli attributi chiave sono evidenziati sottolineandone il nome oppure colorando il cerchietto dell'attributo.

Entità deboli e forti

Entità deboli e forti

- L'**associazione** tra le entità **Persona** e **Automobile**, è stata introdotta l'entità **Acquisto** senza indicare una **chiave primaria.**
- Entità come **Acquisto**, che **non hanno una chiave primaria** e necessitano di essere associate a un'altra entità per essere completamente significative, prendono il nome di **entità deboli.**



Entità deboli e forti

Entità come **Persona** e **Automobile che**

→ sono **completamente identificabili** mediante gli attributi che le caratterizzano

→ hanno una **chiave primaria**

prendono il nome di **entità forti.**

Presa singolarmente, l'entità Acquisto non ha molto significato a differenza delle entità Persona e Automobile.

Entità deboli e forti

- Riflettendo sul significato degli **attributi** dell'entità **Acquisto** si nota che, presi singolarmente, **nessuno di essi** può fungere da **chiave primaria**.
- Questo è ovvio per **DataAcquisto** e **PrezzoAcquisto**, mentre per **Targa** bisogna ricordare che si considerano gli acquisti eseguiti presso una concessionaria.
- In tale caso è possibile che lo stesso veicolo possa essere venduto più volte, prima come veicolo nuovo e in seguito come usato.
- Neanche l'attributo Targa può essere usato come chiave primaria.

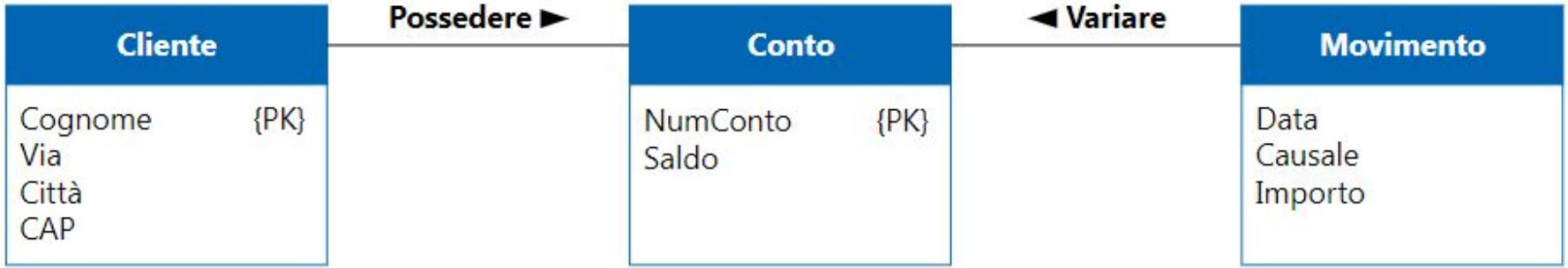
Chiavi composte

- La precedente osservazione permette di concludere che la coppia di attributi (**DataAcquisto, Targa**) identifica univocamente ogni singolo acquisto e la **chiave primaria** di Acquisto è una **chiave composta**, formata dalla coppia di **chiavi parziali** DataAcquisto e Targa (PPK, Partial Primary Key).

Con questa precisazione, si può dire che *Acquisto* non è un'entità debole.

Entità deboli e forti - esempio “conto corrente”

- Lo schema **E/R seguente**, che contiene l’entità **Movimento** con gli attributi Data, Causale e Importo, ha le medesime caratteristiche degli attributi di Acquisto.
- Movimento **non ha chiave primaria** e ha significato solo in associazione all’**entità Conto**.



Tornando alle entità deboli e forti → Aggiunta di contatore

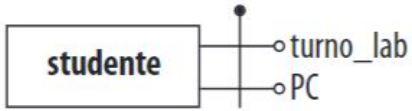
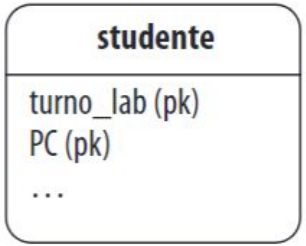
- Per evitare di avere **entità deboli** in alternativa si può aggiungere un attributo di tipo **numero progressivo** assegnato automaticamente, che identifica un'istanza dell'entità.

Ogni movimento risulta univocamente identificato dal valore di questo attributo, che, pertanto, costituisce anche la chiave primaria dell'entità.

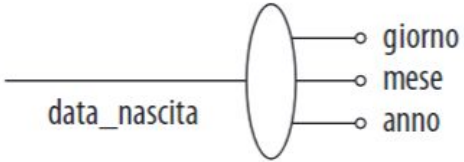
- Questo modo di operare, inoltre, permette di ordinare cronologicamente i movimenti.
- Diversamente i movimenti risulterebbero indistinguibili dal punto di vista temporale.

Chiavi - confronto tra notazione classica e UML

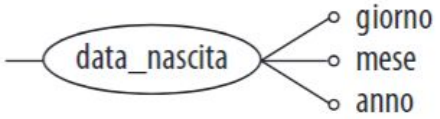
Nel caso di chiavi composte ci sono più possibili notazioni grafiche, oltre ad aggiungere la sigla pk a entrambe quella più utilizzata è riportata nell'esempio seguente:

NOTAZIONE CLASSICA	NOTAZIONE UML
	

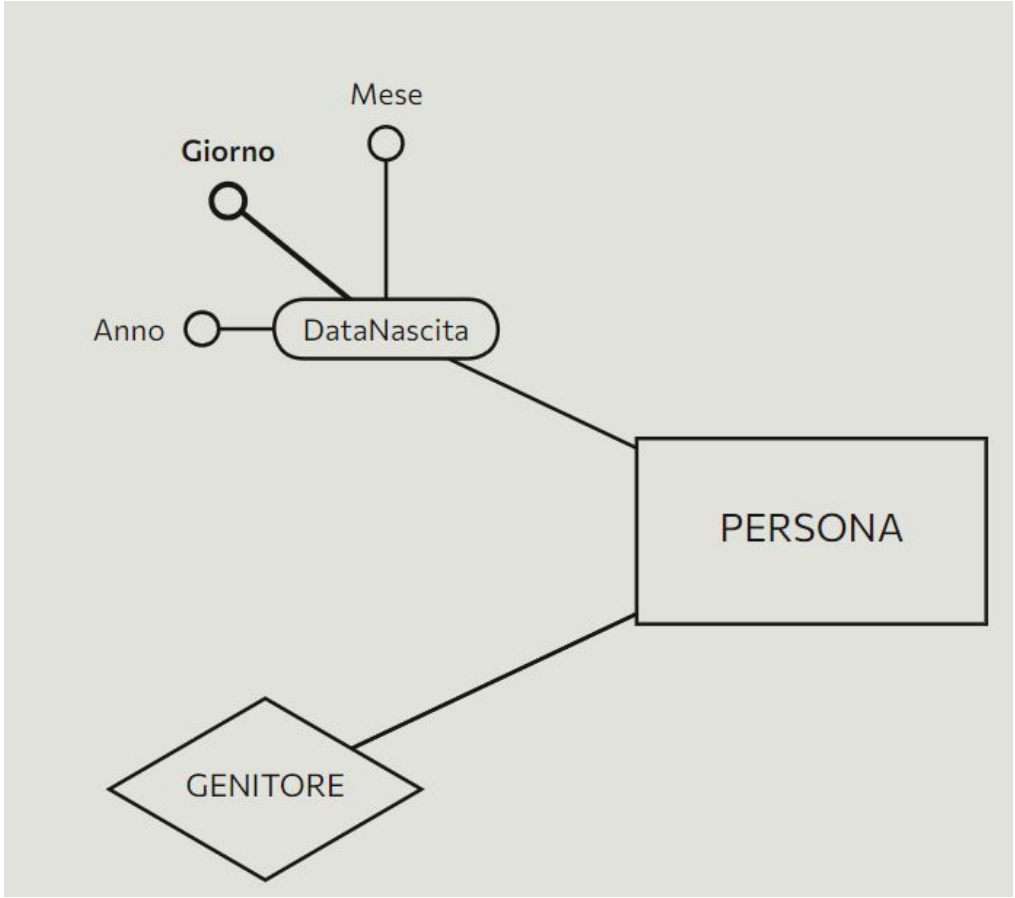
Gli attributi composti, come telefono, indirizzo, data_nascita, vengono rappresentati nella notazione classica secondo una delle due notazioni grafiche di seguito riportate.



oppure



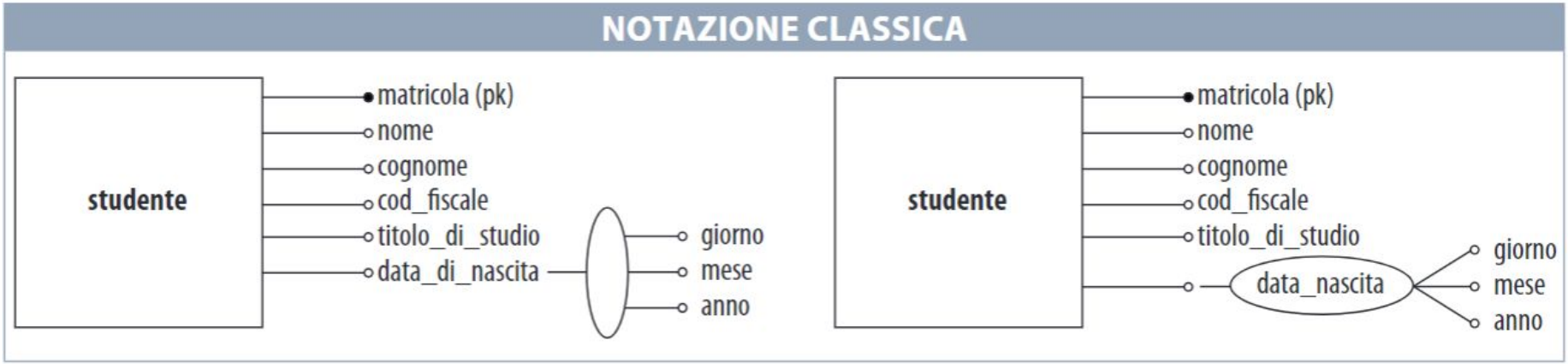
Chiavi - confronto tra notazione classica e UML



https://designer-basic.polito.it/

Le entità - confronto tra notazione classica e UML

Gli attributi composti, come telefono, indirizzo, data_nascita, vengono rappresentati nella notazione classica secondo una delle due notazioni grafiche di seguito riportate.



Nella notazione UML non è prevista la rappresentazione degli attributi composti.

Molteplicità delle relazioni

Le associazioni (relazioni) tra entità

- La **molteplicità** di un'associazione è il **numero di possibili istanze** di un'**entità**, messo in corrispondenza con un'**istanza dell'altra entità che partecipa all'associazione.**

Il **numero minimo** e **massimo** di possibili **istanze** viene rappresentato mediante una coppia di valori separati da punti: 1..1, 0..1, 1..N.

Al valore **minimo** e **massimo** sono associati gli importanti concetti di **obbligatorietà** e **cardinalità** dell'associazione.

Le associazioni (relazioni) tra entità

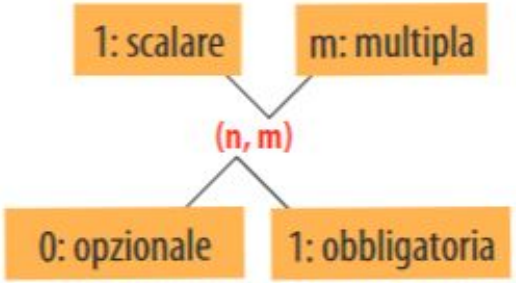
Al valore **minimo** e **massimo** sono associati gli importanti concetti di **obbligatorietà** e **cardinalità** dell'associazione:

- il **valore minimo** assume, in genere, uno dei due valori 0 e 1.
  - 0 indica che la partecipazione è **facoltativa**
  - 1 indica che la partecipazione è **obbligatoria**;
- il **valore massimo** definisce la **cardinalità** della **partecipazione** all'associazione.

Esso assume in genere uno dei due valori 1 oppure N per indicare **una** o **molte** partecipazioni all'associazione.

Cardinalità delle relazioni - notazione classica

La cardinalità minima e massima viene indicata sul diagramma con (n, m) dove la prima cifra indica l'esistenza e la seconda la molteplicità.

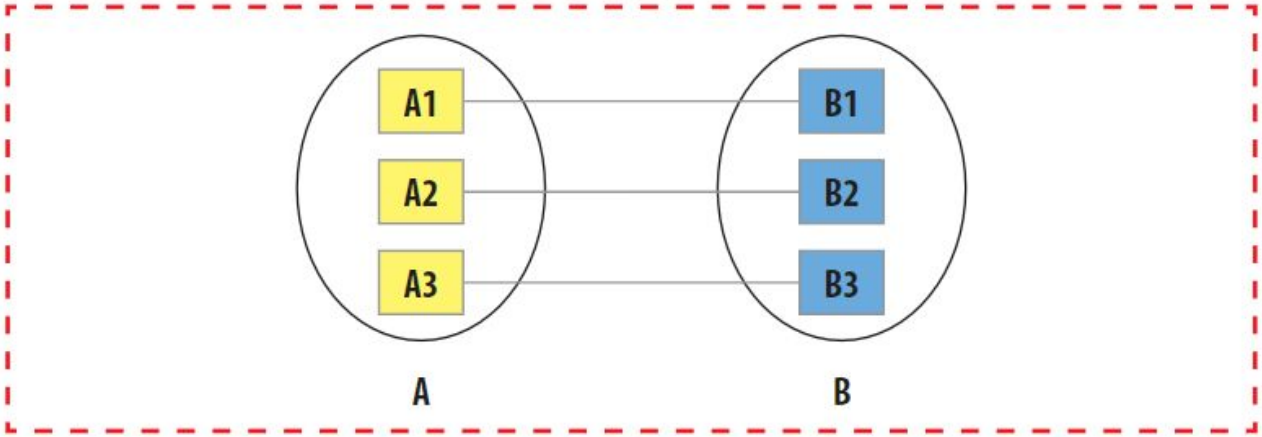


L'esistenza (o minima cardinalità) di un'entità in una relazione indica che può essere obbligatoria oppure opzionale:

- 0 opzionale;
- 1 obbligatoria.

Le relazioni di tipo 1:1

Date due entità **A** e **B**, la relazione uno-a-uno (1, 1) si ottiene quando al massimo una istanza dell'entità **A** viene associata a una sola istanza dell'entità **B**.



Le relazioni di tipo 1:1

Un primo esempio di relazione uno-a-uno potrebbe essere il seguente:

> *Gli impiegati di una società dispongono di un proprio ufficio personale.*

Per ogni **impiegato** esiste cioè un unico **ufficio** e per ogni ufficio esiste un unico **impiegato**.

Un secondo esempio è il seguente:

> *Ogni nazione ha una capitale.*

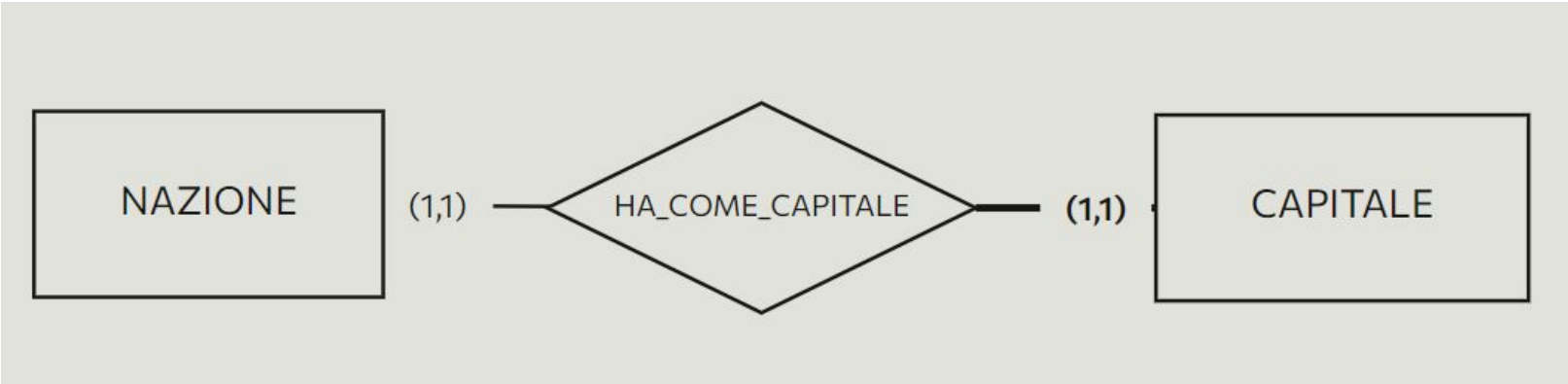
Esiste cioè un'unica **capitale** per ogni **nazione** e una città può essere **capitale** di una sola **nazione**.

Vediamo ora un ultimo esempio.

> *Ogni persona ha una sua impronta digitale.*

Esiste cioè un'unica **impronta digitale** per ogni **persona** e un'**impronta digitale** è specifica di una sola **persona**.

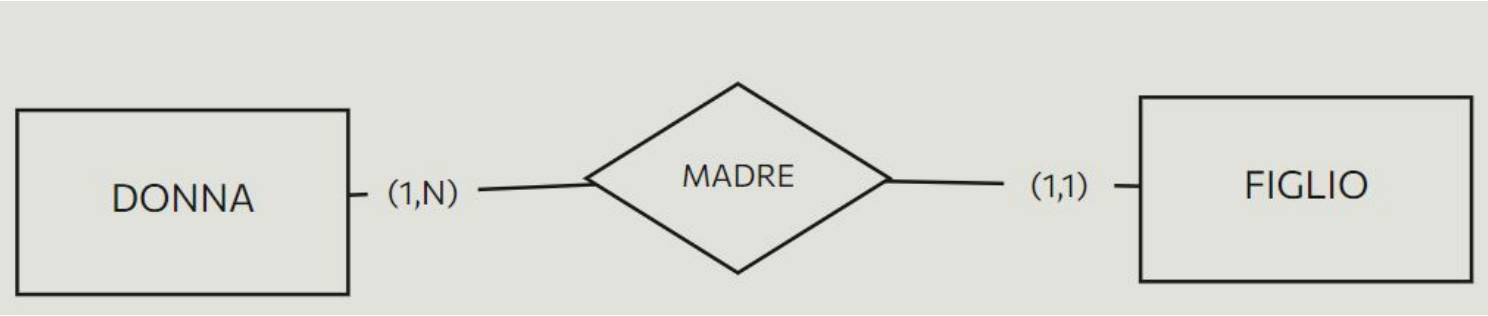
Le relazioni di tipo uno a uno (1:1)



- Nazione **ha come capitale** una sola città (1-->1)
  - partecipazione obbligatoria
  - molteplicità 1
- Un capitale **appartiene** ad una sola nazione (1 ←1)
  - partecipazione obbligatoria
  - molteplicità 1

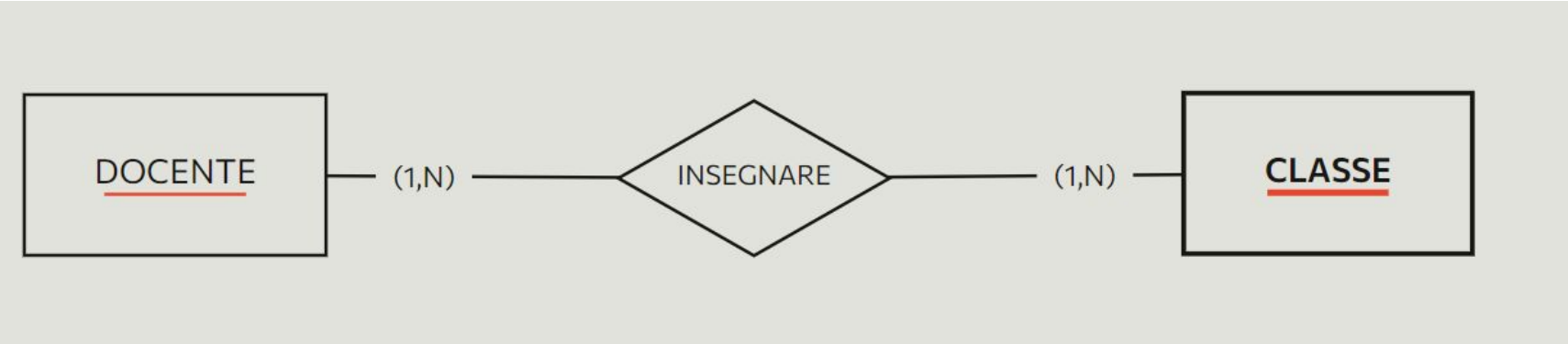
Relazioni uno a molti

- Una relazione si dice **uno-a-molti** se esiste un'istanza della prima entità cui corrisponde **più di** un'istanza della seconda ma a ogni istanza della seconda entità corrisponde **al più** un'istanza della prima entità
- viene anche indicata con **(1, n)**;



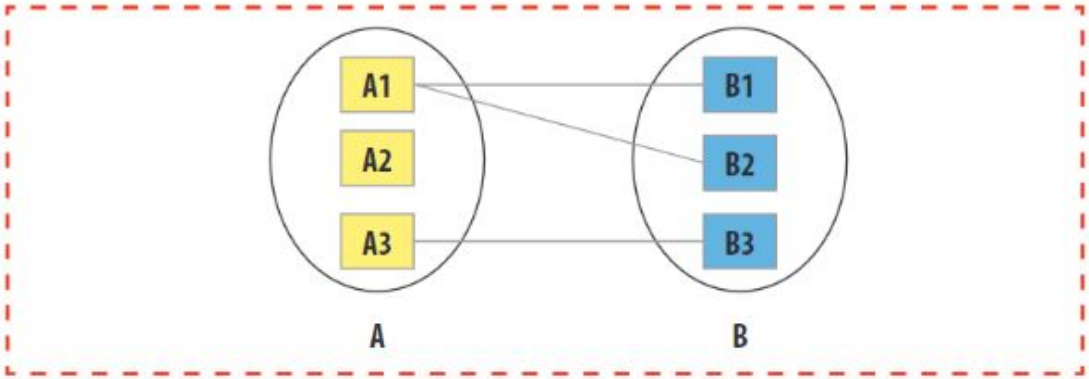
Relazioni di tipo molti a molti

- Una relazione si dice **molti-a-molti** se esiste un'istanza della prima entità in relazione con più di un'istanza della seconda, e viceversa: viene indicata con **(n, n)**



Le relazioni molti-a-molti

La relazione uno-a-molti (1, n) si ha quando per una istanza dell'entità **A** ci sono zero, una, o molte istanze dell'entità **B**, ma per una istanza dell'entità **B** c'è solo una istanza dell'entità **A**.



- Un ufficio ha molti impiegati, e ogni impiegato è associato a un solo ufficio.
- Una scuola ha molti alunni, e ogni alunno frequenta una scuola.
- In una città abitano molte persone, e ogni persona abita in una città.
- Uno studente abita in una città, in una città abitano diversi studenti.
- Una donna ha più figli, un figlio ha una sola madre.

Esistenza obbligatoria

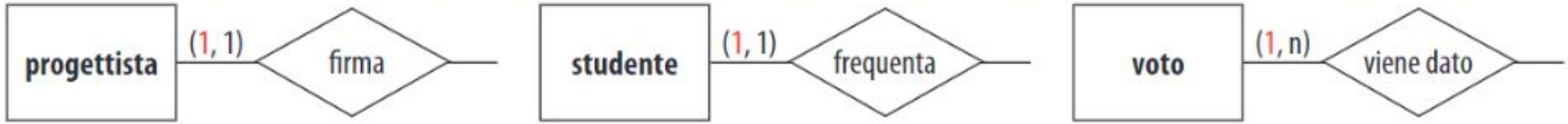
Se una istanza di un'entità deve necessariamente esserci perché un'entità sia inclusa in una relazione, allora l'esistenza è obbligatoria.

Ecco alcuni esempi di esistenza obbligatoria.

- Un **voto** **deve** essere attribuito a uno **studente** (non ha senso un voto non attribuito).
- Uno **studente** **deve** frequentare un **corso** (non esistono studenti senza corsi di studio).
- Ogni **progetto** **deve** essere firmato da un **tecnico** (non possono esistere progetti senza il progettista).

Osserviamo come le entità sono "legate" dalla forma verbale **deve**.

NOTAZIONE CLASSICA



Esistenza opzionale

Fattura-Ordine

- A ogni **fattura** può essere associato **nessuno** (0) oppure **un ordine**.
- A ogni **ordine** è associata **una sola fattura** (1).

- L’esistenza si dice opzionale se un istanza di un'entità può partecipare facoltativamente alla relazione.

Molteplicità

La molteplicità (o massima cardinalità) indica il numero massimo di istanze che partecipano alla relazione:

- 1: una istanza;
- <un valore>: un numero massimo di istanze;
- n: senza limiti.

A ogni impiegato è assegnato sempre almeno un incarico e, al massimo, cinque incarichi (5), mentre un dato incarico può essere non assegnato a nessuno o, comunque, al massimo a cinquanta impiegati (50).

Molteplicità

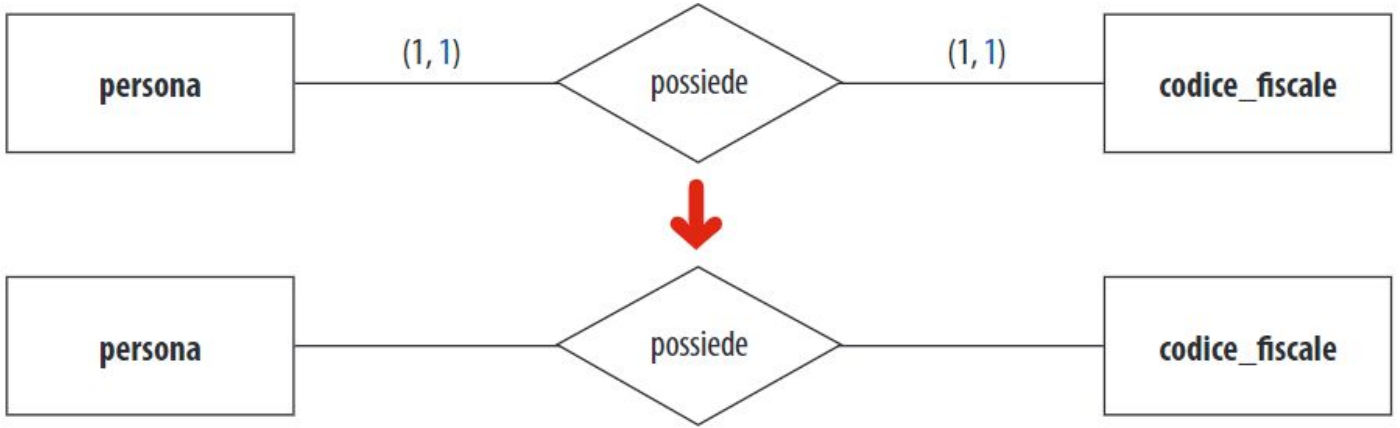
Nel caso di relazione **scalare (1) obbligatoria (1)**, generalmente il numero viene omesso sul diagramma, come nell'esempio che segue

Altri esempi di molteplicità delle relazioni

Esempi di vincoli di cardinalità

Uno-a-uno (1, 1)

La relazione è uno-a-uno se la cardinalità massima di entrambe le entità è 1.



- Una **persona** possiede **almeno** 1 e **massimo** 1 codice fiscale (-->)
- Un **codice_fiscale** appartiene ad almeno una ed al massimo una persona (<--)

Esempi di vincoli di cardinalità

Uno-a-molti (1, n)

La relazione è uno-a-molti se la massima cardinalità verso una entità è 1 e la massima cardinalità verso l'altra entità è n.

- Una persona risiede in almeno **1** e al massimo **1** città (-->)
- Un città risiedono almeno **1** e al massimo **n** persone(<--)

Altri esempi

Persona-Automobile

- minima cardinalità (**persona**, **proprietario**) = 0: esistono persone che non posseggono alcuna automobile;
- massima cardinalità (**persona**, **proprietario**) = n: ogni persona può essere proprietaria di un numero arbitrario di automobili;
- minima cardinalità (**automobile**, **proprietario**) = 0: esistono automobili non possedute da alcuna persona;
- massima cardinalità (**automobile**, **proprietario**) = 1: ogni automobile può avere al più un proprietario.

Quindi lo schema completo di vincoli è il seguente:

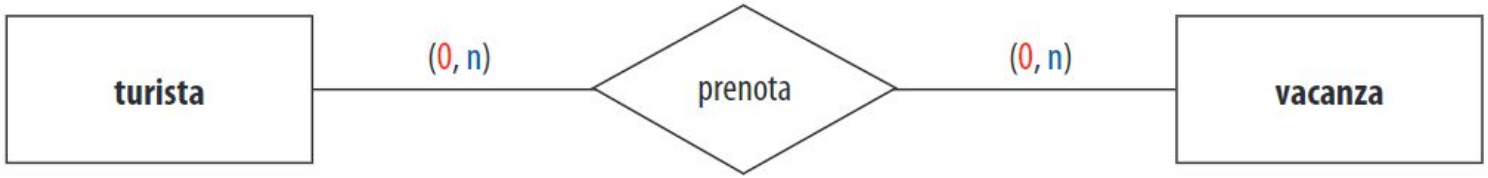
Altri esempi

Persona-Comune

- a ogni **persona** viene associato (1) un solo (1) **comune** di nascita;
- a ogni **comune** possono essere associate da 0 a n nascite di **persone**.

Turista-Vacanza

- ogni **turista** può effettuare nessuna (0) oppure alcune (n) prenotazioni di una **vacanza**.
- a ogni **vacanza** possono essere associate nessuna (0) a tante (n) prenotazioni di **turisti**.



Direzione della relazione

Direzione della relazione

È anche possibile completare il diagramma E-R introducendo un simbolo grafico che indica la direzione della relazione, nel caso questa sia definita.

La direzione di una relazione indica l'entità da cui trae origine la relazione binaria: l'entità da cui si parte si chiama entità padre e l'entità a cui si arriva si chiama entità figlio.

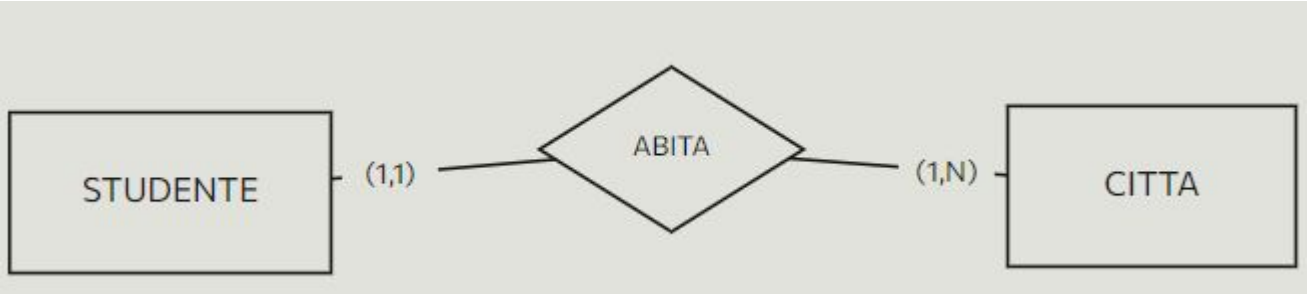
La direzione di una relazione viene determinata dalla sua cardinalità:

- in relazioni di tipo uno-a-uno la direzione è dall'entità forte a quella debole; se entrambe sono forti, la direzione è arbitraria;
- nelle relazioni uno-a-molti, l'entità con cardinalità uno è il padre della relazione con cardinalità molti;
- la direzione nelle relazioni molti-a-molti è arbitraria.

Direzione della relazione

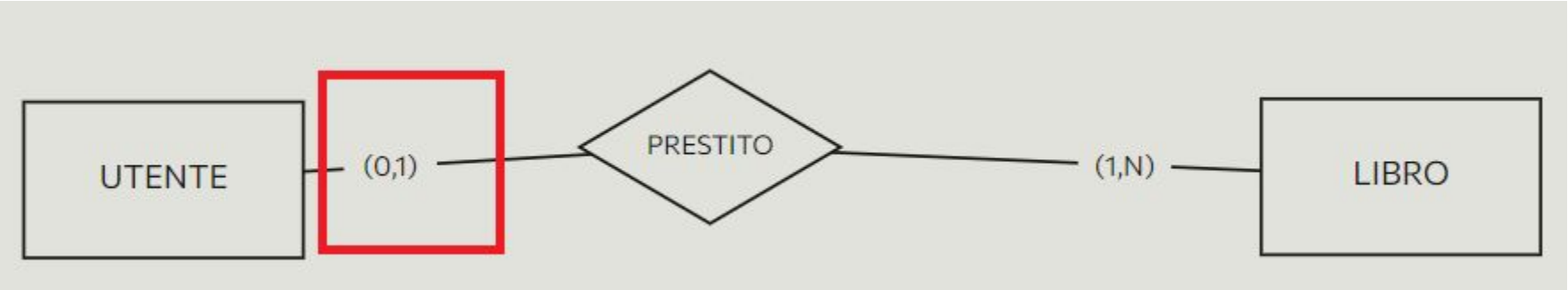
- Riconsiderando l'esempio delle entità **studente** e **città**, si vede che, poiché la relazione tra città e studente ha cardinalità **uno-a-molti**, la **direzione** è da città a studente. →

- L'entità **città** è padre rispetto all'entità studente



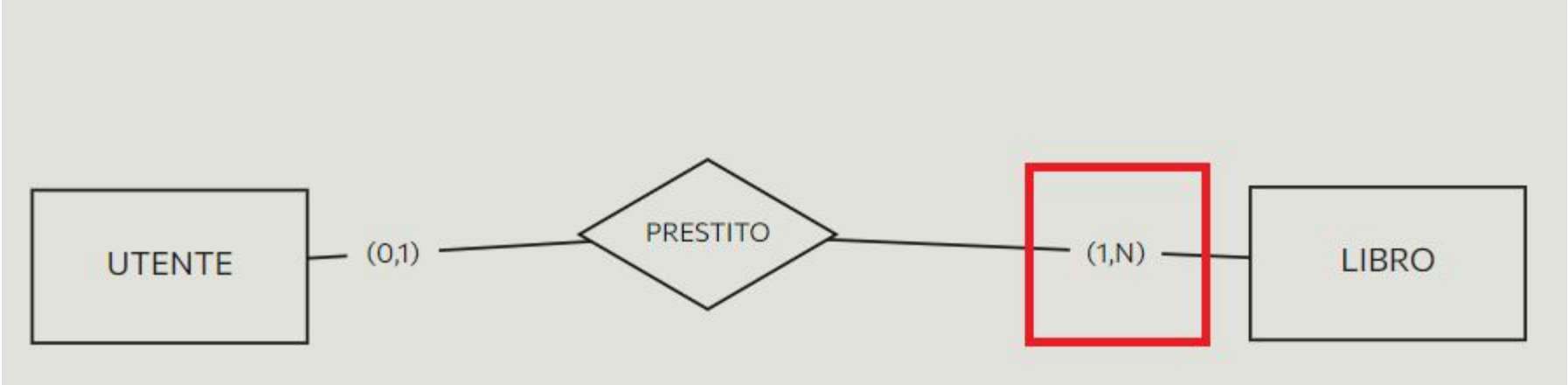
Le regole di lettura

- Entità di partenza: **Utente**
- **Verso di lettura**: da sinistra verso destra
- (0,1) si legge
  - (0, → **può** chiedere in prestito
  - ,1) → **un solo** libro



Le regole di lettura

- Entità di partenza: **Libro**
- **Verso di lettura**: da destra verso sinistra
- (1,N) si legge
  - (1, → un libro **deve** essere prestato
  - ,N) → **uno o più** utenti



Relazione gerarchica tra entità

Relazione gerarchica tra le entità

Esistono situazioni in cui tra le **entità** può essere stabilita una **gerarchia**, come nella gerarchia delle classi della programmazione **OOP (Object-Oriented Programming)**.

Osserviamo la seguente situazione riportata a lato:

- **beta** è detta **sottoclasse** o **specializzazione** di alfa;
- **alfa** è detta **superclasse** o **generalizzazione** di beta

Relazione gerarchica tra le entità

Nelle gerarchie sono presenti **due vincoli** (o proprietà):

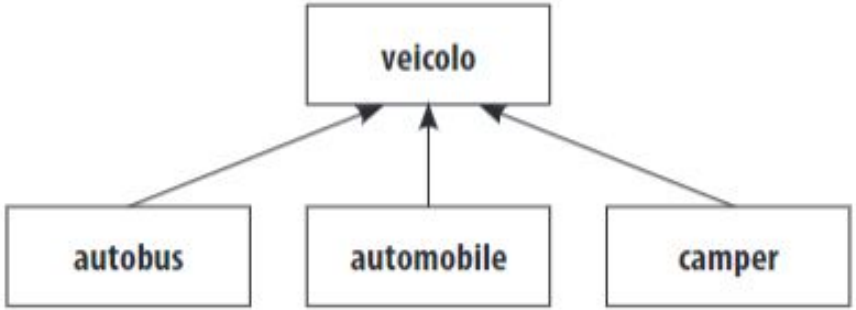
1. **vincolo di struttura**: se beta è **sottoclasse** di alfa, beta ha tutti gli **attributi** di alfa e **partecipa a tutte le associazioni** cui partecipa alfa (ereditarietà); questo non esclude che beta possa avere altri attributi e partecipare ad altre associazioni.

2. **vincolo di insieme**: se beta è **specializzazione** di alfa, ogni oggetto di beta è anche un oggetto di alfa (cioè beta è un sottoinsieme di alfa).

Nella fase di progettazione dello schema E-R capita sovente di definire entità che hanno tra loro **dipendenze gerarchiche**.

Relazione gerarchica tra le entità

Le istanze di **automobile** sono un sottoinsieme delle istanze di **veicolo**, ovvero, ogni automobile è un (is a) veicolo.



Ciò che caratterizza un **veicolo** caratterizza anche ogni suo sottoinsieme, ovvero ogni sottoclasse eredita dalla superclasse, ma può anche avere caratteristiche proprie.

Relazione gerarchica tra le entità

Le “**sottoentità**” individuano un gruppo di elementi della classe **padre**, cioè **specializzano** e individuano un **sottoinsieme** di elementi: non è detto tuttavia che l’unione delle parti contempli tutta la casistica degli elementi della classe padre.

Nell’esempio precedente dei veicoli non esiste infatti nessuna sottoentità che individua i trattori oppure i camion ecc.

Per indicare queste situazioni si introduce il concetto di **copertura** delle generalizzazioni, che può essere **totale** o **parziale**, **esclusiva** o **sovrapposta.**

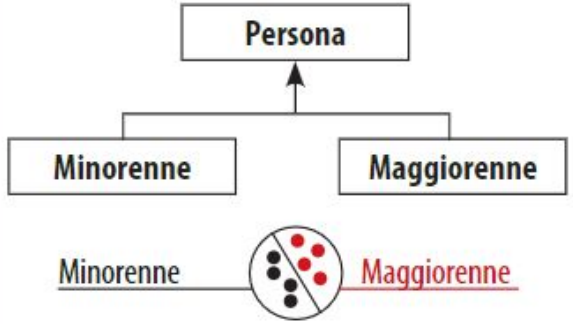
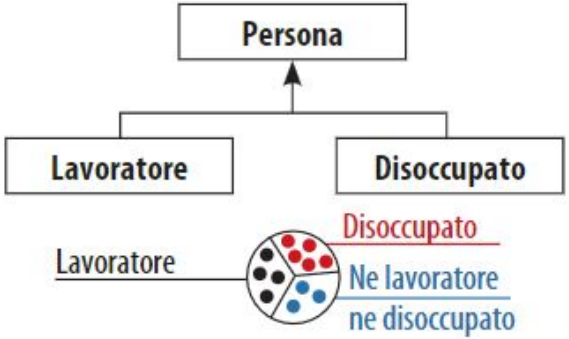
Relazione gerarchica tra le entità

Le **generalizzazioni** si caratterizzano quindi per “due dimensioni indipendenti”:

1. confronto fra **unione delle specializzazioni** e classe generalizzata:
   a. **totale**, se la classe generalizzata **è** l’unione delle specializzazioni;
   b. **parziale**, se la classe generalizzata **contiene** l’unione delle specializzazioni;
2. confronto fra le classi specializzate:
   a. **esclusiva**, se le specializzazioni sono fra loro **disgiunte**;
   b. **sovrapposta** (overlapped), se può esistere un’intersezione non vuota fra le specializzazioni.

Relazione gerarchica tra le entità

Vediamo attraverso un esempio le quattro possibili combinazioni ottenibili da esse.

	Totale Ogni oggetto della superclasse appartiene a una sottoclasse	**Parziale** Possono esistere oggetti della superclasse che non sono in alcuna sottoclasse
Esclusiva Le sottoclassi non hanno oggetti in comune		
Sovrapposta Le sottoclassi possono avere oggetti in comune