

Basi di Dati

UniShare

Davide Cozzi
@dlcgold

Gabriele De Rosa
@derogab

Federica Di Lauro
@f_dila

Indice

1	Introduzione	2
2	Introduzione al Corso	3
3	Progettazione Concettuale:ER	11
3.0.1	Relazione IS-A	25

Capitolo 1

Introduzione

Questi appunti sono presi a ldurante le esercitazioni in laboratorio. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: <https://github.com/dlcgold/Appunti>.

Grazie mille e buono studio!

Capitolo 2

Introduzione al Corso

Il corso di Basi di Dati affronta gli aspetti e i metodi per lo sviluppo di un database in maniera efficiente, aspetto fondamentale per un informatico e per lo sviluppo ottimale di software.

Il corso si divide in 7 parti:

1. introduzione generale
2. metodologie e modelli per il progetto delle basi di dati
3. progettazione concettuale
4. modello razionale
5. progettazione logica
6. linguaggio SQL
7. algebra relazionale

Le **informazioni** fanno parte delle risorse di un'azienda, soprattutto negli ultimi anni di clima globale, per cui la gestione efficiente ed ottimale dei dati è fondamentale, come ad esempio Facebook ed Amazon fanno ampio uso dei nostri dati, sia a scopi pubblicitari sia a scopi di marketing.

Un sistema informativo è una componente di un'organizzazione che gestisce le informazioni d'interesse, non per forza attraverso un'automatizzazione e/o supporto di un calcolatore, infatti sin dall'antichità le banche tenevano traccia dei depositi tramite un archivio cartaceo.

Una porzione automatizzata del sistema informativo si chiama sistema informatico, che si divide in:

- acquisizione e memorizzazione

- aggiornamento
- interrogazione
- elaborazione

Le informazioni vengono gestite in vari modi, attraverso il linguaggio naturale, graficamente con schemi e/o numeri, con il tempo si è arrivati a codifiche standard per quasi tutte le tipologie di informazioni, e sono rappresentate nei sistemi informatici dai *dati*, la cui differenza è che i dati sono valori senza alcun valore mentre le informazioni stabilisce un'interpretazione attribuendo una semantica ai valori.

I dati sono una risorsa strategica in quanto sono stabili nel tempo, infatti solitamente i dati sono immutati durante una migrazione tra un sistema e un altro, per questo lo sviluppo progettazione di un database rimane stabile in teoria, senza notevoli cambiamenti durante la durata di un sistema informativo.

Un **Data Base** è una collezione di dati usati per rappresentare le informazioni di interesse di un sistema informativo, definite solo una volta a cui un insieme di applicazioni ed utenti può accedere ad essi, mentre un **DBMS** è un software per la gestione di un database.

I dati presenti in un database sono molti, indipendenti dal programma in cui vengono utilizzati e cui si cerca di evitare la ridondanza, per garantire la consistenza delle informazioni.

Per la creazione di un database, al fine di garantire privacy, affidabilità, efficienza ed efficacia, sono presenti le seguenti tre fasi:

1. definizione
2. creazione e popolazione
3. manipolazione

Un'organizzazione è divisa in vari settori e ogni settore ha un suo sottosistema informativo, non necessariamente disgiunto, e i database solitamente sono condivisi, al fine di ridurre la ridondanza delle informazioni, per cui sono presenti meccanismi di autorizzazione e di controllo della concorrenza.

Un database deve essere conservato a lungo termine e si ha una gestione delle **transazioni**: insieme di operazioni da considerare indivisibile, atomico, corretto anche in presenza di concorrenza e con effetti definitivi.

La sequenza di operazioni nel database deve essere eseguita nella sua interezza, per cui l'effetto di transazioni concorrenti deve essere coerente, infatti la conclusione positiva di una transazione corrisponde ad un impegno,

commit, a mantenere traccia del risultato, anche in presenza di guasti e di esecuzioni concorrente.

Come tutti i software, i DBMS devono essere efficienti, utilizzando al meglio memoria e tempo, efficaci e produttivi.

Si hanno delle caratteristiche nell'approccio alla base dati:

- natura autodescrittiva di un sistema di basi di dati: il sistema di basi di dati memorizza i dati con anche una descrizione completa della sua struttura (metadati), per consentire ai DBMS di lavorare con qualsiasi applicazione.
- separazione tra programmi e dati, infatti è possibile cambiare la struttura dati senza cambiare i programmi.
- astrazione dei dati: si usa un modello dati per nascondere dettagli e presentare all'utente una visione concettuale del database.
- supporto di viste multiple dei dati, per cui ogni utente può usare una vista (view) differente del database, contenente solo i dati di interesse per quell'utente.
- condivisione dei dati e gestione delle transazioni con utenti multipli.

I DBMS estendono le funzionalità dei file system, fornendo più servizi ed in maniera integrata.

In ogni base di dati si ha:

- lo **schema**, sostanzialmente invariante nel tempo, che ne descrive la struttura, l'aspetto intensionale.
- l'**istanza**, i valori attuali, che possono cambiare anche molto rapidamente, l'aspetto estensionale "concreto".

Per lo sviluppo delle basi di dati si hanno due tipi di modelli, ambedue importanti:

- **modelli logici**, adottati nei DBMS esistenti per l'organizzazione dei dati, utilizzati dai programmi e sono indipendenti dalle strutture fisiche
- **modelli concettuali** che permettono di rappresentare i dati in modo indipendente da ogni sistema, con il fine di descrivere i concetti del mondo reali; sono usati nelle fasi preliminari di progettazione e il più diffuso è il modello **Entity-Relationship**.

Un database è organizzato solitamente attraverso i tre schemi dell'architettura ANSI/SPARC:

- schema logico: descrizione dell'intera base di dati nel modello logico “principale” del DBMS, ossia si definisce la struttura concettuale del database, senza considerare l'implementazione fisica nel DBMS.
- schema fisico: rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione
- schema esterno: descrizione di parte della base di dati in un modello logico (“viste” parziali, derivate, anche in modelli diversi)

L'accesso ai dati avviene solo mediante il livello esterno, il quale a volte coincide con quello logico, e si hanno 2 forme di indipendenza: quella fisica, in cui è possibile interagire con il DBMS senza conoscere la struttura fisica e quella logica, in cui si può accedere al livello esterno senza interagire con lo schema logico.

Per la definizione dei database ci sono quattro tipologie di linguaggi:

- **DLL**(Data Manipulation Languages) linguaggio per definire i dati
- **DML**(Data Manipulation Languages) linguaggio per la manipolazione dei dati
- **DCL**(Data Control Languages) linguaggio per il controllo degli accessi al database
- **DQL**(Data Query Languages) linguaggio per effettuare delle interrogazioni al database

Noi vedremo SQL per definire i database, linguaggio basato sull'algebra relazionale che implementa tutti e 4 le tipologie di linguaggi per la gestione di un database.

Si hanno due tipi di utenti:

1. **utenti finali (terminalisti)**: eseguono applicazioni predefinite (transazioni)
2. **utenti casuali**: eseguono operazioni non previste a priori, usando linguaggi interattivi

inoltre si hanno:

- progettisti e realizzatori di DBMS

- progettisti della base di dati e amministratori della base di dati (DBA), persona o gruppo di persone responsabile del controllo centralizzato del database, in tutti i suoi aspetti.
- progettisti e programmatori di applicazioni



Come visto anche nel corso di analisi e progettazione del software e nella figura X, durante lo sviluppo di un sistema informatico si hanno le seguente fasi:

- **studio di fattibilità**, definizione di costi, priorità e competenze per un progetto
- **raccolta e analisi dei requisiti**, ovvero lo studio delle proprietà del sistema
- **progettazione** di dati e funzioni
- **implementazione**
- **validazione e collaudo**, che comprendono anche test da parte del cliente
- **funzionamento**, ovvero lo stadio finale dove il sistema diventa effettivamente operativo

In questo corso ci occuperemo soltanto delle fasi di progettazione ed implementazione di base di dati, lasciando l'analisi delle altre fasi al corso ed ai libri di Ingegneria del Software.

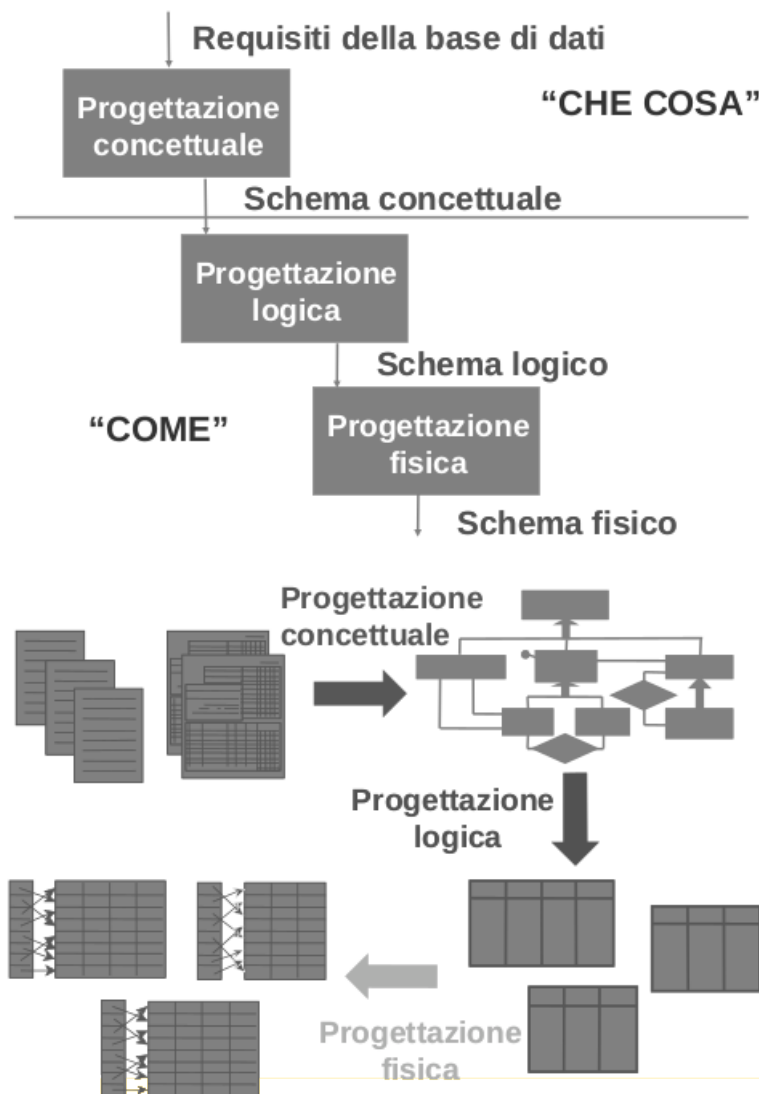
In questo processo a rimanere stabili sono prettamente i **dati** infatti prima si progetta la base dati, con una **metodologia di progetto**, e poi il software.

Ciclo di vita (modello a spirale)



Una **metodologia** è un'articolazione in fasi di guida ad un'attività di progettazione, in caso di una base dati la metodologia, con il fine di separare il cosa rappresentare dal come, è la seguente:

- suddivida la progettazione in fasi indipendenti
- fornisca strategie e criteri di scelta in caso di alternative
- fornisca modelli di riferimento (i linguaggi)
- garantisca generalità rispetto al problema
- garantisca qualità e facilità d'uso



Come si può notare nella figura X1, la metodologia corrente di modellazione di un database prevede la definizione e l'esecuzione delle seguenti fasi:

- la **progettazione concettuale** consiste nel tradurre i requisiti del sistema informatico in una descrizione formale, integrata e indipendente dalle scelte implementative (DBMS, SW e HW)
- la **progettazione concettuale** consiste nella traduzione dello schema concettuale nel modello dei dati scelto per la modellazione, ottenendo uno schema logico, espresso nel DDL del DBMS. In questa fase si considerano anche aspetti legati ai vincoli ed all'efficienza. Si hanno due sotto-fasi:

- ristrutturazione dello schema concettuale
- traduzione verso il modello logico
- la **progettazione fisica** completa lo schema logico ottenuto con le specifiche proprie del DBMS scelto. Il risultato è lo schema fisico che descrive le strutture di memorizzazione ed accesso ai dati

Incominciamo nel prossimo capitolo a considerare la progettazione concettuale, per poi analizzare nei successivi capitoli in dettaglio anche la fase di progettazione concettuale e il linguaggio SQL.

Capitolo 3

Progettazione Concettuale:ER

Come abbiamo già introdotto nel precedente capitolo, affrontiamo ora la progettazione concettuale di una base dati attraverso il modello **ER**(Entity Relationships), modello concettuale che fornisce una serie di strutture atte a descrivere in maniera semplice e facile la realtà di interesse da modellare.

Si hanno dei vantaggi con la progettazione concettuale, prevale infatti l'aspetto intensionale indipendente dalla tecnologia ed è una rappresentazione grafica ed è utile per la documentazione, in quanto facilmente comprensibile anche da persone poco avvezze alla tecnologia e ai database.

In uno schema ER si hanno i seguenti costrutti, come si nota nella figura Y, la cui rappresentazione effettiva varia in quanto vi sono più versioni di ER:

- **entità:** classe di oggetti con proprietà comune ed esistenza "autonoma", della quale si vogliono specificare fatti specifici;ogni entità ha un nome univoco, espressivo e al singolare.
- **relazione:** rappresentano legami logici, significativi per la realtà da modellare, tra due o più entità.
Un'occorrenza di relazione è un n-upla costituita da occorrenze di entità, una per ciascuna delle entità coinvolte, e ogni relazione ha un nome univoco, in cui è preferibile assegnare un sostantivo per evitare di stabilire un verso alla relazione.
Essendo una relazione matematica tra le entità coinvolte non è possibile avere delle n-uple identiche, con conseguenze per la realtà da rappresentare, per esempio non è possibile attraverso una relazione il fatto che è possibile ripetere un esame, obbligando a rappresentare l'esame come entità e non più come relazione.

- **attributo semplice:** associa ad ogni istanza di entità o associazione un valore, definito su un dominio di valori, specificato nella documentazione associata, con il fine di descrivere le proprietà elementari di entità e/o relazioni disegnate per rappresentare la realtà d'interesse.
- **attributo composto:** raggruppamento di attributi di una medesima entità/relazione con affinità di significato e/o uso come ad esempio possiamo raggruppare gli attributi Via, Numero Civico e Cap dall'entità persona per formare l'attributo composto Indirizzo.
- **cardinalità delle relazioni:** vengono specificate per ogni relazione e descrivono il numero minimo e massimo di occorrenze di relazione, a cui una occorrenza dell'entità può partecipare alla relazione, ossia quante volte un'occorrenza di un'entità può essere legata ad occorrenze delle altre entità coinvolte.
È possibile assegnare un qualunque intero non negativo, con l'unico vincolo che la cardinalità minima sia minore o uguale alla cardinalità massima e di solito si usano i valori $0, 1eN$, indicanti zero, una o molte occorrenze, senza preoccuparsi in caso di N del numero effettivo di occorrenze.
- **cardinalità di un attributo:** descrivono il numero minimo e massimo di valori dell'attributo associati all'entità e/o relazione, con la cardinalità $(1, 1)$ stabilita come default, che può essere vista come funzione che associa ad ogni occorrenza di entità un solo valore dell'attributo; si hanno le stesse consuetudini delle cardinalità delle relazioni.
- **identificatore interno:** permette di identificare in maniera univoca un'entità ed un identificatore è interno in caso sia uno o più attributi di un'entità, tutti con cardinalità $(1, 1)$.
- **identificatore esterno:** un identificatore è esterno, in caso un'entità E viene identificata da un'attributo di un'entità F , cui esiste una relazione uno a uno tra l'entità E e F .
È possibile, anche se molto raro, avere la definizione dell'identificatore usando entità di entità, ossia l'identificatore dell'entità E viene definito nell'entità G , cui esiste una relazione con l'entità F che a

sua volta ha una relazione con l'entità E , ma si può capire da quanto è contorto il ragionamento qual'è la sua percentuale d'utilizzo nella modellazione dello schema ER.




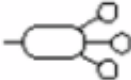



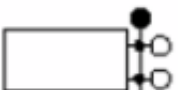
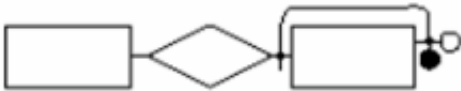


- **generalizzazione:** rappresentano legami logici tra un'entità E , detta padre, e una serie di entità E_1, E_2, \dots, E_n , dette figlie, di cui l'entità E rappresenta un caso generale della serie di entità figlie.

Tra le entità coinvolte in una generalizzazione valgono le seguenti proprietà:

- ogni occorrenza di un'entità figlia è anche un'occorrenza dell'entità genitore.
- ogni proprietà dell'entità genitore è anche una proprietà delle entità figlie, quindi nello schema ER non devono essere rappresentate, e ciò prende il nome di ereditarietà.

Le generalizzazioni possono essere classificate sulla base di due proprietà ortogonali:

- una generalizzazione è totale se ogni occorrenza del genitore è un'occorrenza di almeno uno dei figli, altrimenti è parziale.
 - una generalizzazione è esclusiva se ogni occorrenza del genitore è al più un'occorrenza di una delle entità figlie, altrimenti è sovrapposta.
- **sottoinsieme:** generalizzazione con soltanto un'entità figlia, di cui solitamente rappresenta una parte dell'entità genitore come ad esempio gli studenti sono un sottoinsieme delle persone.

Construct	Graphical representation
Entity	
Relationship	
Simple attribute	
Composite attribute	
Cardinality of a	
Cardinality of an attribute	
Internal identifier	 
External identifier	
Generalization	
Subset	

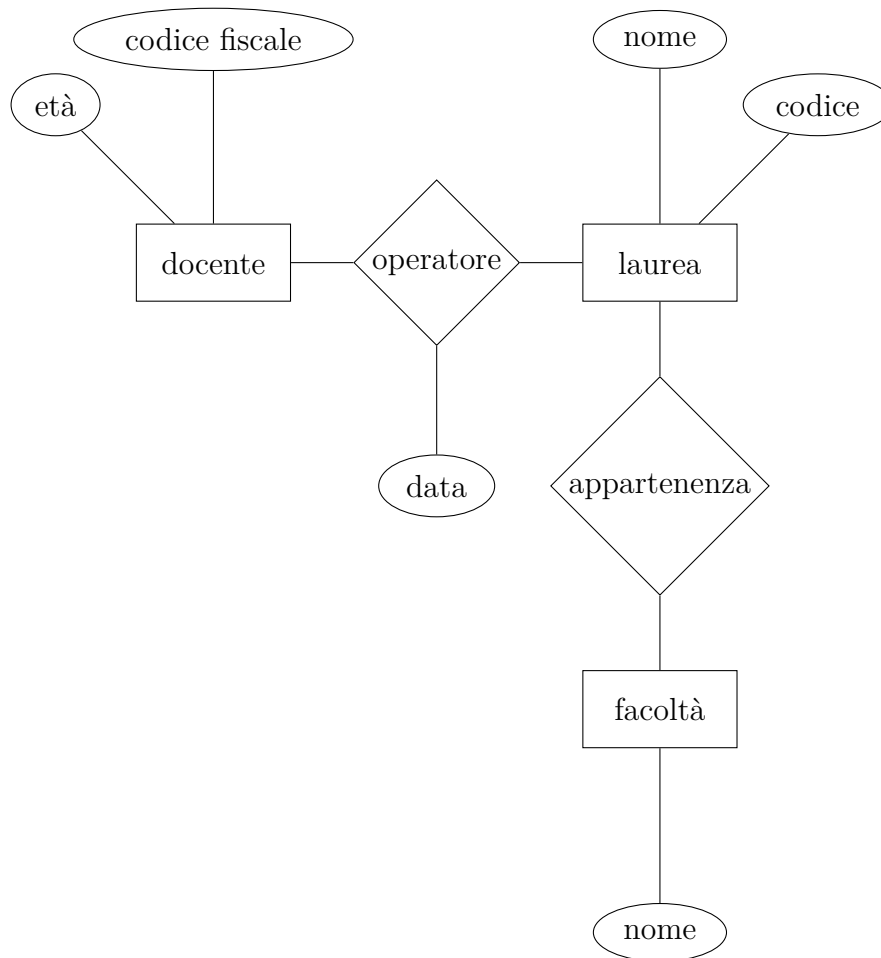
Un'occorrenza, o istanza, di un'entità, è un oggetto della classe che l'entità rappresenta, ma noi rappresentiamo le entità dello schema concettuale non le singole istanze: si parla di *conoscenza astratta* se si parla di entità e di *conoscenza concreta* se si parla di un'istanza di un'entità. affinità nel loro significato o uso (per esempio per un indirizzo si ha via, comune, cap etc...). Nessuno impone un tipo per un certo attributo, possono essere anche complessi di cui però non ci interessano le informazioni che lo rappresentano (per esempio una foto può essere un attributo, ma se ho bisogno, per esempio, dell'autore della foto, non sarà più un attributo ma un'altra entità).

Un'**istanza di associazione** è una combinazione o aggregazione di istanze di entità che prendono parte all'associazione (per esempio "prof. Schettini")

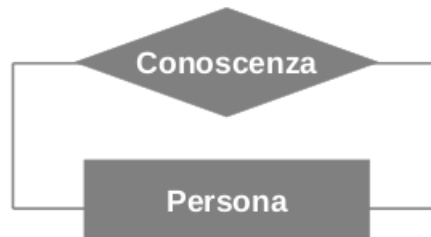
è istanza di associazione per l'entità docente).

Le relazioni possono avere attributi, con valori specificati in un certo dominio, che modella una proprietà del legame tra tutte le entità rappresentato dalla relazione.

Descrivere lo schema concettuale della seguente realtà: I docenti hanno un codice fiscale ed una età. I docenti operano nei corsi di laurea (si dice che afferiscono ai corsi di laurea). Interessa la data di afferenza dei docenti ai corsi di laurea. I corsi di laurea hanno un codice ed un nome, ed appartengono alle facoltà. Ogni facoltà ha un nome



Una associazione può coinvolgere “due o più volte” la stessa entità. Si ha un’*associazione ricorsiva o ad anello*:

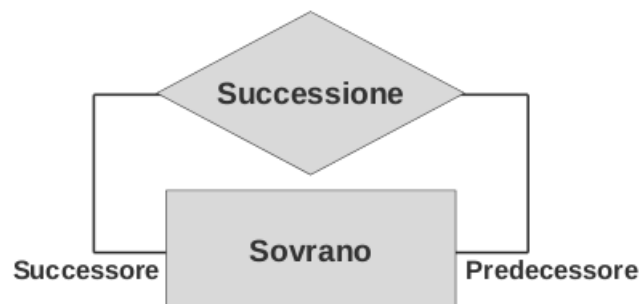


Un'associazione ad anello può essere o meno:

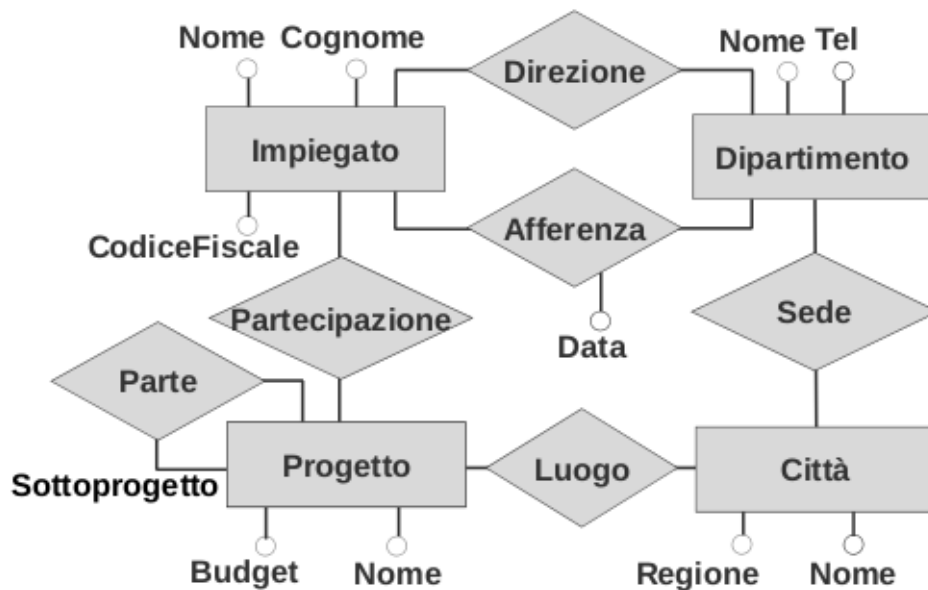
- **simmetrica:** $(a, b) \in A \rightarrow (b, a) \in A$
- **riflessiva:** $(a, a) \in A$
- **transitiva:** $(a, b) \in A, (b, c) \in A \rightarrow (a, c) \in A$

Nel caso sopra l'associazione conoscenza è simmetrica, irriflessiva e intransitiva.

Nelle relazioni dove una stessa entità è coinvolta più volte è necessario aggiungere la specifica dei **ruoli**, come nell'esempio:



Esempio 1. Descrivere lo schema concettuale della seguente realtà: Degli impiegati interessa il codice fiscale, il nome, il cognome, i dipartimenti ai quali afferiscono (con la data di afferenza), ed i progetti ai quali partecipano. Dei progetti interessa il nome, il budget, e la città in cui hanno luogo le corrispondenti attività. Alcuni progetti sono parti di altri progetti, e sono detti loro sottoprogetti. Dei dipartimenti interessa il nome, il numero di telefono, gli impiegati che li dirigono, e la città dove è localizzata la sede. Delle città interessa il nome e la regione:



Si sceglie di modellare:

- un'entità:
 - se le sue istanze sono concettualmente significative indipendentemente da altre istanze
 - se ha o potrà avere delle proprietà indipendenti dagli altri concetti
 - se il concetto è importante nell'applicazione
- un attributo:
 - se le sue istanze non sono concettualmente significative
 - se non ha senso considerare una sua istanza indipendentemente da altre istanze

- se serve solo a rappresentare una proprietà locale di un altro concetto

Si sceglie di modellare:

- un'entità:
 - se le sue istanze sono concettualmente significative indipendentemente da altre istanze
 - se ha o potrà avere delle proprietà indipendenti dagli altri concetti
 - se ha o potrà avere relazioni con altri concetti
- una relazione:
 - se le sue istanze non sono concettualmente significative indipendentemente da altre istanze, cioè se le sue istanze rappresentano n-ple di altre istanze
 - se non ha senso pensare alla partecipazione delle sue istanze ad altre relazioni

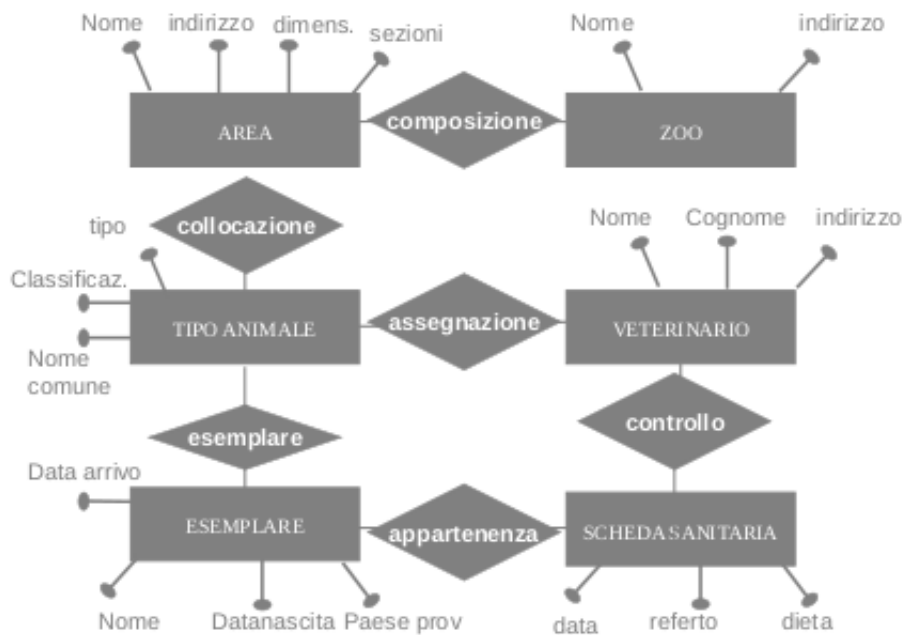
per esempio:

•Calcio: un giocatore gioca in una squadra

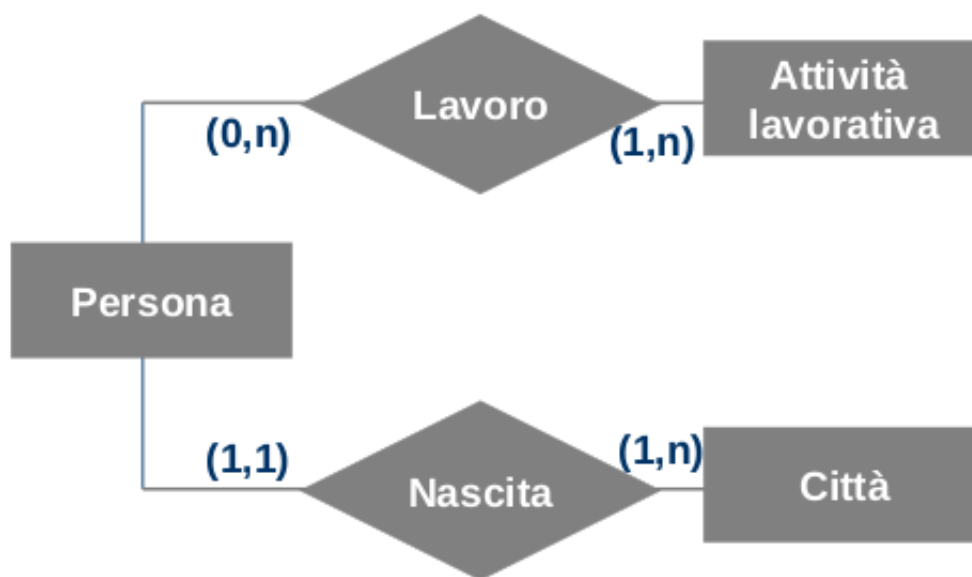


qui ruolo è attributo della composizione perché un giocatore può cambiare ruolo e quindi ruolo non è attributo di giocatore.

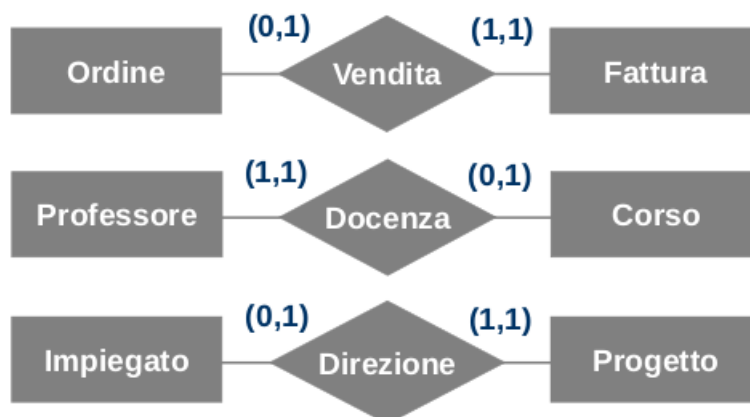
- Esempio 2.**
- ogni zoo è diviso in aree diverse a seconda che si tratti di rettili, pesci, uccelli, scimmie, grandi mammiferi, ... Ogni area è dotata di: nome, indirizzo, dimensione, numero di sezioni.
 - per ogni tipo di animale ci sono informazioni che riguardano: classificazione zoologica, nome comune (giraffa, elefante, serpente, tartaruga, ...), habitat, alimentazione, ... Per ogni tipo di animale c'è un diverso veterinario specialista, dipendente dello zoo.
 - ogni tipo di animale è rappresentato da esemplari e relativi dati anagrafici: nome proprio (giraffa Enrico, giraffa Giulia, ...), data di nascita, Paese di provenienza, data di arrivo allo zoo, ...
 - ogni esemplare è dotato di più schede sanitarie contenenti ognuna: la data della visita, referto, dieta, nome del veterinario, ...



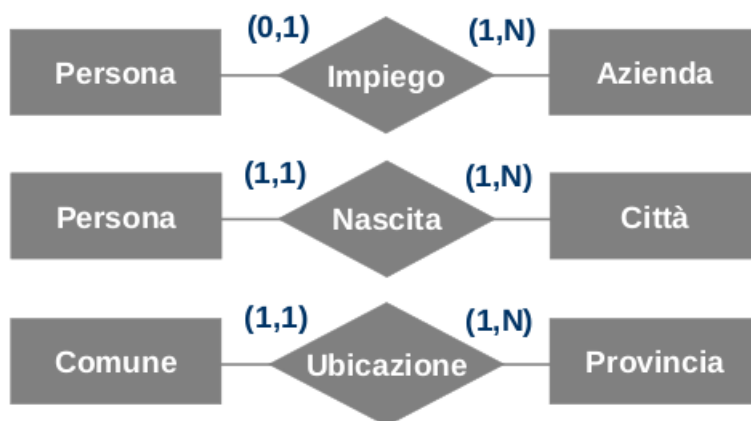
lo zoo ha degli attributi che vanno indicati anche se non richiesti dalla traccia per evitare ambiguità. Dato che ogni scheda è collegata ad un veterinario, di cui ho un'entità la collego a veterinario con una relazione e non ne faccio un attributo



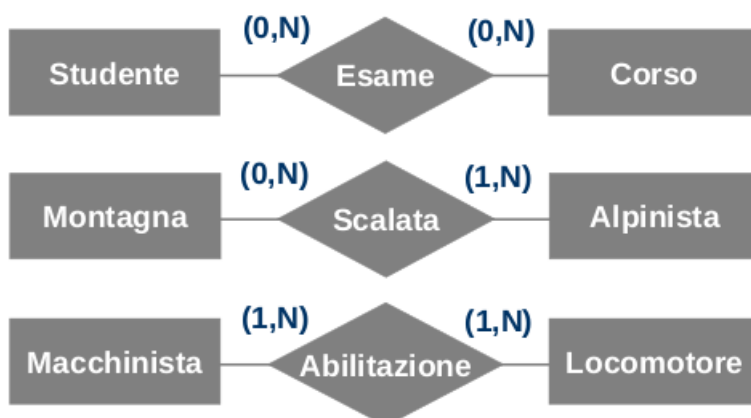
Relazioni "uno a uno"



Relazioni “uno a molti”



Relazioni “molti a molti”



vediamo un esempio:

Esempio 3. *si ha:*



- $\text{min-card}(\text{Automobile}, \text{Proprietario}) = 0$: esistono automobili non possedute da alcuna persona
- $\text{min-card}(\text{Persona}, \text{Proprietario}) = 0$: esistono persone che non posseggono alcuna automobile
- $\text{max-card}(\text{Persona}, \text{Proprietario}) = n$: ogni persona può essere proprietaria di un numero arbitrario di automobili
- $\text{max-card}(\text{Automobile}, \text{Proprietario}) = 1$: ogni automobile può avere al più un proprietario



Istanza dello schema:

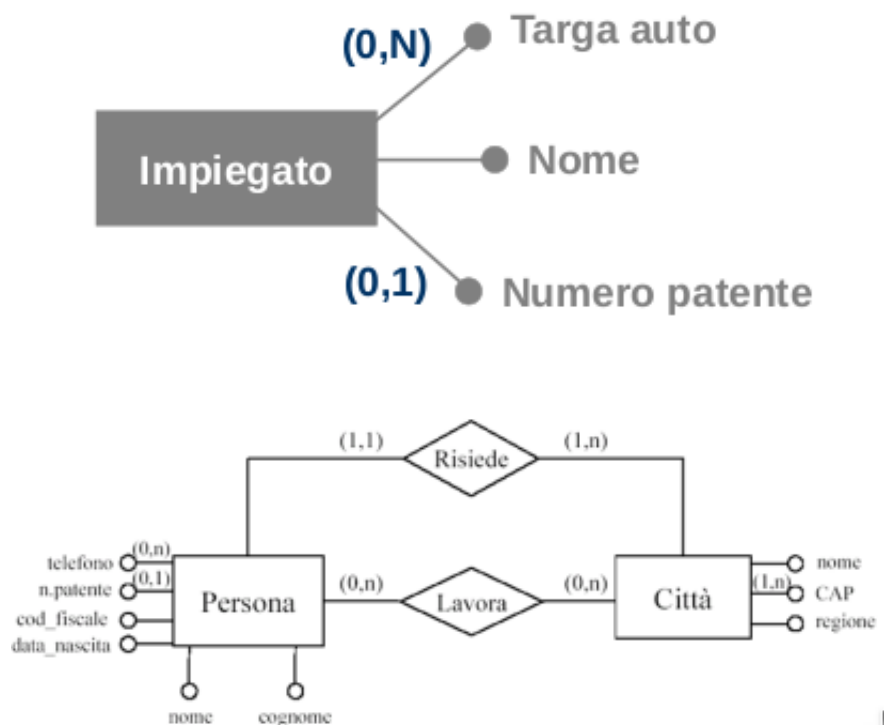
Istanze(Impiegato) = { a,b,c }

istanze(Progetto) = { x,y,v,w,z }

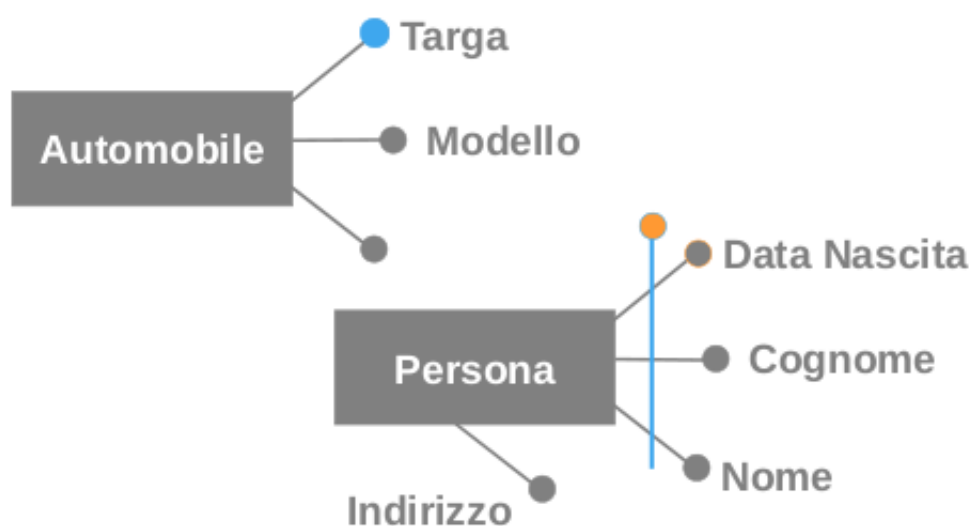
istanze(Assegnazione) = { (a,w), (b,v), (b,w), (c,y), (c,w), (c,z) }

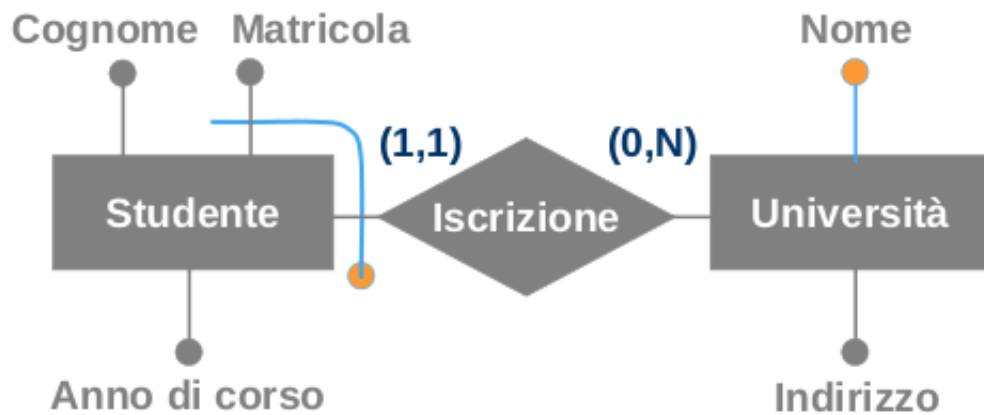
Ad ogni impiegato sono assegnati da 1 a 5 progetti e ogni progetto è assegnato ad al più 50 impiegati. a,b,c compaiono in almeno una istanza di Assegnazione. x non compare nelle istanze di Assegnazione. Infine ci sono progetti (ad esempio lanciati da poco tempo) che possono non essere assegnati a nessun impiegato

Si può mettere cardinalità agli attributi per indicare l'opzionalità o indicare attributi multivalore:

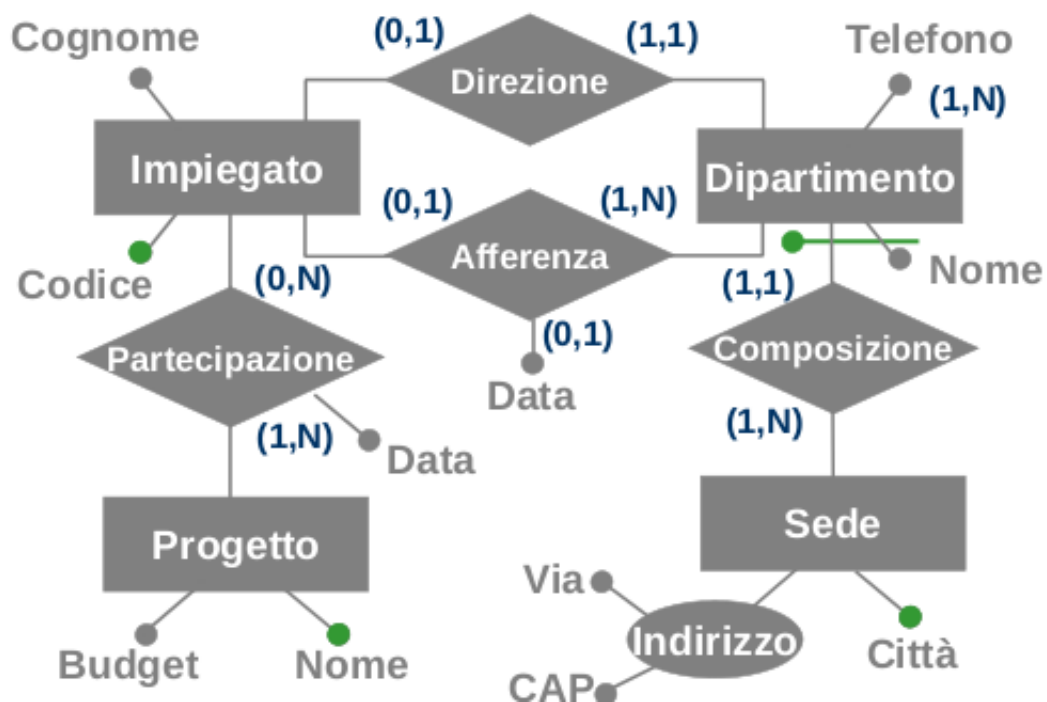


infatti, per esempio, una persona può non avere la patente etc...





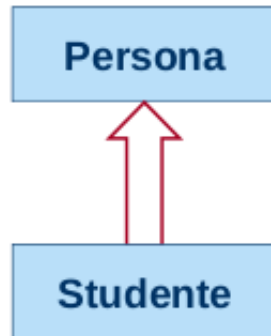
vediamo un esempio complesso, che è preso da un vecchio esempio:



3.0.1 Relazione IS-A

È facile verificare che, in molti contesti, può accadere che tra due classi rappresentate da due entità nello schema concettuale sussista la relazione IS-A (o relazione di sottoinsieme), e cioè che ogni istanza di una sia anche istanza dell'altra. La relazione IS-A nel modello ER si può definire tra due

entità, che si dicono *entità padre* ed *entità figlia* (o sottoentità, cioè quella che rappresenta un sottoinsieme della entità padre). La relazione IS-A si rappresenta con una freccia dalla sottoentità alla entità padre:



Si ha il **principio di ereditarietà**: *ogni proprietà dell'entità padre è anche una proprietà della sottoentità, e non si riporta esplicitamente nel diagramma. L'entità figlia può avere ovviamente ulteriori proprietà.*

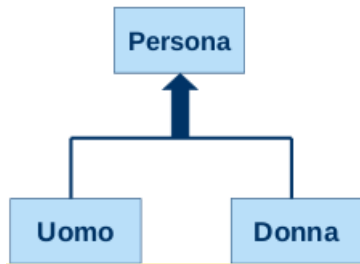
Finora, abbiamo considerato la relazione ISA che stabilisce che l'entità padre è più generale della sottoentità. Talvolta, però, l'entità padre può generalizzare diverse sottoentità rispetto ad un unico criterio. In questo caso si parla di **generalizzazione**. Nella generalizzazione, le sottoentità hanno insiemi di istanze disgiunti a coppie (anche se in alcune varianti del modello ER, si può specificare se due sottoentità della stessa entità padre sono disgiunte o no). Una generalizzazione può essere di due tipi:

1. **completa**: l'unione delle istanze delle sottoentità è uguale all'insieme delle istanze dell'entità padre
2. **non completa**

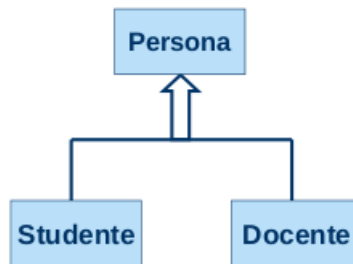
Si dice che:

- E è **generalizzazione** di E_1, \dots, E_n
- E_1, \dots, E_n sono **specializzazioni (o sottotipi)** di E

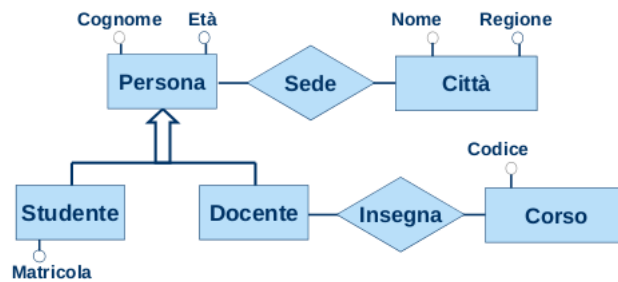
La generalizzazione si indica collegando mediante un arco le sottoentità, e collegando con una freccia tale arco alla entità padre. La freccia è annerita se la generalizzazione è completa:



La freccia non è annerita se la generalizzazione non è completa:



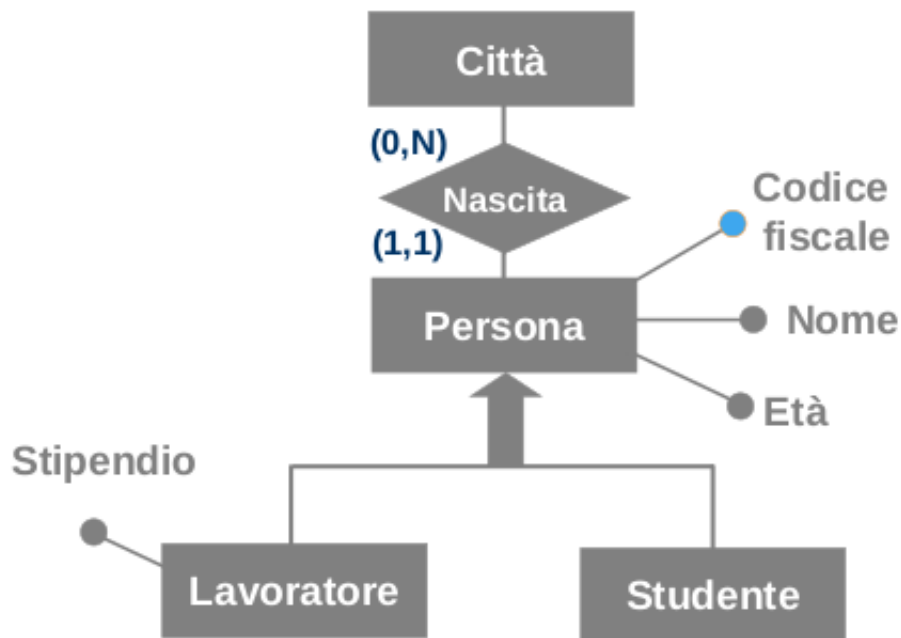
Il principio di ereditarietà vale anche per le generalizzazioni: *ogni proprietà dell'entità padre è anche una proprietà della sottoentità, e non si riporta esplicitamente nel diagramma. L'entità figlia può avere ovviamente ulteriori proprietà:*



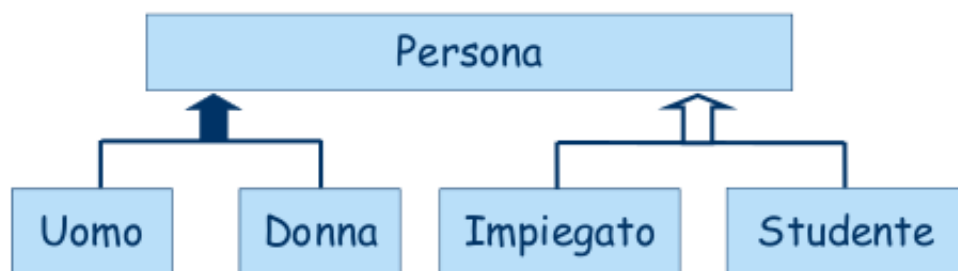
Si hanno delle proprietà per le generalizzazioni. Se E , il genitore, è generalizzazione delle figlie E_1, \dots, E_n si ha:

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

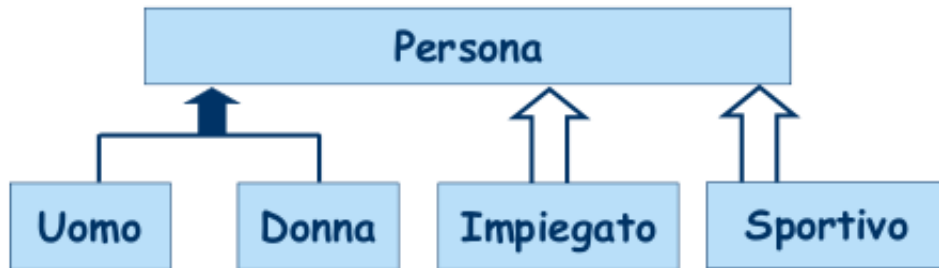
per esempio:



La stessa entità può essere padre in diverse generalizzazioni. Concettualmente, non c'è alcuna correlazione tra due generalizzazioni diverse, perché rispondono a due criteri diversi di classificare le istanze della entità padre:



Si possono avere comunque sottoentità indipendenti, nel senso che il loro significato non deriva dallo stesso criterio di classificazione delle istanze della entità padre:



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

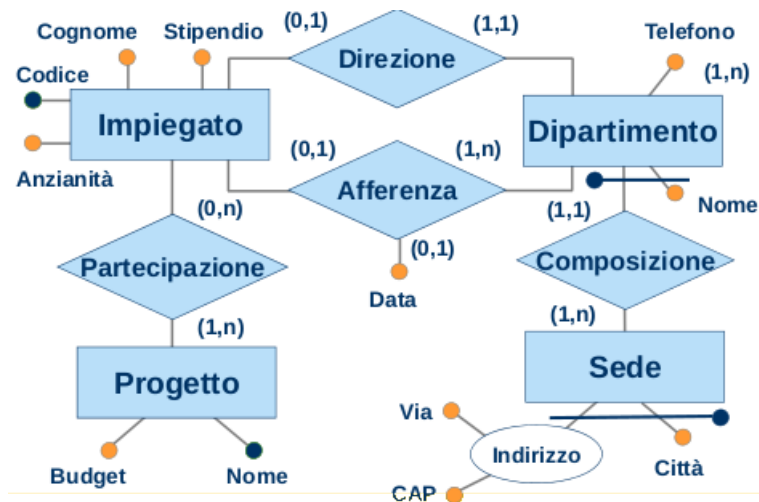
Si hanno anche altre proprietà:

- possono esistere gerarchie a più livelli e multiple generalizzazioni allo stesso livello
- un'entità può essere inclusa in più gerarchie, come genitore e/o come figlia
- se una generalizzazione ha solo un'entità figlia si parla di sottoinsieme

Alcune interrelazioni non possono essere colte direttamente da uno schema ER. Tali interrelazioni vanno in ogni caso tenute presenti attraverso delle asserzioni aggiuntive dette vincoli esterni al diagramma, o semplicemente vincoli esterni. Questi vincoli possono essere rappresentati con:

- formalismi opportuni (es, in logica matematica)
- asserzioni in linguaggio naturale (che devono essere il più possibile precise e non ambigue)

vediamo un esempio. Si ha il seguente schema concettuale:



con dei vincoli esterni:

Vincoli di integrità esterni	
(1)	Il direttore di un dipartimento deve afferire a tale dipartimento da almeno 5 anni
(2)	Un impiegato non deve avere uno stipendio maggiore del direttore del dipartimento al quale afferisce
(3)	Un dipartimento con sede a Roma deve essere diretto da un impiegato con più di dieci anni di anzianità
(4)	Un impiegato non può partecipare ad un numero di progetti maggiore di due volte il numero di dipartimenti ai quali afferisce

si ottiene:

