Basi di dati

spitfire

A.A. 2023-2024

Contents

1 Introduzione

Una base di dati è un **insieme organizzato di dati** utilizzati per il supporto allo svolgimento di attività (di un ente, azienda, ufficio, persona). Facciamo un passo indietro e ricordiamo cos'è invece **l'informatica**: essa è la **scienza del trattamento razionale dell'informazione**, spesso per mezzo di **macchine automatiche**. Essa è considerata come supporto alla conoscenza umana e alla comunicazione. L'informatica ha due anime:

- **Metodologica**: i metodi per la soluzione di problemi e la gestione delle informazioni
- **Tecnologica**: i calcolatori elettronici e i sistemi che li utilizzano

1.1 Sistemi informativi

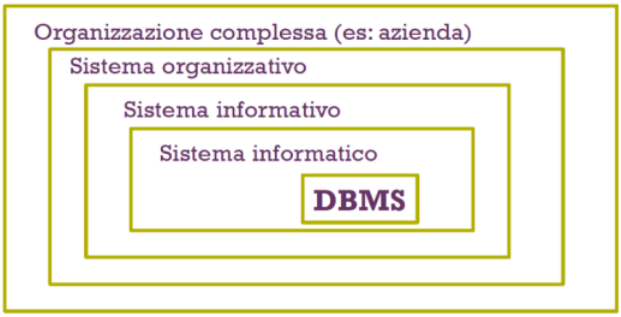
Un **sistema informativo** è un componente (sottosistema) di un'organizzazione che gestisce le informazioni di interesse (cioè utilizzate per il perseguimento degli scopi dell'organizzazione). Le funzioni di un sistema informativo sono:

- **Acquisizione/memorizzazione**
- **Aggiornamento**
- **Interrogazione**
- **Elaborazione**

Il concetto di "sistema informativo" è **indipendente da qualsiasi automazione**: esistono organizzazioni la cui ragion d'essere è la **gestione di informazioni** (es. servizi anagrafici e banche) e che operano da secoli. Anche prima di essere automatizzati, molti sistemi informativi si sono evoluti verso una **razionalizzazione e standardizzazione** delle procedure e dell'organizzazione delle informazioni.

1.2 Sistema informatico

Un **sistema informatico** è una porzione **automatizzata** di un sistema informativo; cioè la parte del sistema informativo che gestisce le informazioni con tecnologia informatica.



Un sistema informatico:

- Garantisce che i dati siano conservati in modo **permanente** sui dispositivi di memorizzazione
- Permette un **rapido aggiornamento** dei dati per riflettere rapidamente le loro variazioni
- Rende i dati **accessibili alle interrogazioni** degli utenti
- Può essere **distribuito sul territorio**

1.3 Gestione delle informazioni

Nelle attività umane, le informazioni vengono gestite (registrate e scambiate) in forme diverse:

- **Idee informali**
- **Linguaggio naturale** (scritto o parlato, formale o colloquiale, in una lingua o in un'altra)
- **Disegni, grafici e schemi**
- **Numeri e codici**

E su vari supporti, come la memoria umana, su carta o su dispositivi elettronici. Nelle attività standardizzate dei sistemi informativi complessi, sono state introdotte col tempo forme di organizzazione e codifica delle informazioni via via più precise (e in un certo senso artificiali). Ad esempio, nei servizi anagrafici si è iniziato con registrazioni discorsive per poi evolversi in registrazioni più complete.

1.4 Informazioni e dati

Nei sistemi informatici (e non solo), le **informazioni** vengono rappresentate in modo essenziale, spartano: attraverso i **dati**. Diamo delle definizioni semantiche:

- **Informazione**: notizia, dato o elemento che consente di avere conoscenza più o meno esatta di fatti, situazioni, modi di essere.
- **Dato**: ciò che è immediatamente presente alla conoscenza, prima di ogni elaborazione; (in informatica) elementi di informazione costituiti da simboli che debbono essere elaborati

I dati quindi hanno bisogno di essere interpretati!

1.4.1 Perché i dati?

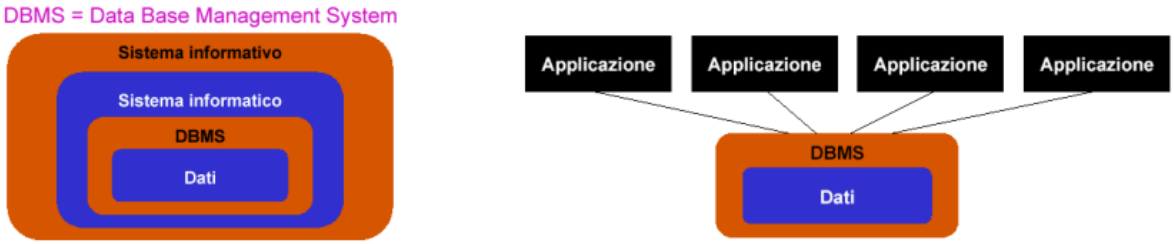
La rappresentazione precisa di forme più ricche di informazione e conoscenza è difficile. I dati costituiscono spesso una risorsa strategica, perché più stabili nel tempo di altri componenti (processi, tecnologie, ruoli, umani). I dati rimangono gli stessi nella **migrazione** da un sistema al successivo.

1.5 Basi di Dati e introduzione ai Database Management Systems (DBMS)

Come già detto, una **base di dati** (DB) è una collezione di dati utilizzati per rappresentare le informazioni di interesse in un sistema informativo. Un **Database Management System** (DBMS) è invece un **sistema software** capace di gestire collezioni di dati che siano *grandi, condivise e persistenti*, assicurando la loro *affidabilità e privatezza*. Possiamo quindi dare due accezioni al termine **base di dati**:

- **Metodologica**: insieme organizzato di dati utilizzati per il supporto allo svolgimento delle attività di un ente
- **Specifica, metodologica e tecnologica**: Insieme di dati gestito da un DBMS

Possiamo dare ancora una definizione: una **base di dati** è un insieme di archivi in cui ogni dato è rappresentato logicamente una sola volta e può essere utilizzato da un insieme di applicazioni o da diversi utenti secondo opportuni criteri di riservatezza.



Le caratteristiche di una base di dati sono:

- I dati sono **molti**
- I dati hanno un **formato definito**
- I dati sono **permanenti**
- I dati sono **raggruppati** per insiemi **omogenei** di dati
- Esistono **relazioni specifiche** tra gli insiemi di dati
- La ridondanza è **minima e controllata**: è assicurata la consistenza delle informazioni
- I dati sono disponibili **per utenze diverse e concorrenti**
- I dati sono **controllati**: protetti da malfunzionamenti hardware e software
- I dati della base di dati sono **indipendenti dai dati di un qualsiasi programma**

1.6 Database Management System

Un DBMS è un insieme di programmi che permettono di creare, usare e gestire una base di dati. Quindi un DBMS è un sistema software general purpose che facilita il processo di **definizione, costruzione e manipolazione** del database per varie applicazioni. Vi sono 3 fasi nella **creazione di un database**:

1. **Definizione**
2. **Creazione/Popolazione**
3. **Manipolazione**

Una volta creato un database, si potranno effettuare su di esso delle **interrogazioni** (queries). L'efficacia delle queries dipende da:

- Conoscenza dle contenuto del DB
- Esperienza con il linguaggio di interrogazione
- Semplicità ed efficacia dell'interfaccia di interrogazione

Un DBMS è quindi un **sistema** che gestisce collezioni di dati:

- **Grandi**
- **Persistenti**
- **Condivise**

garantendo:

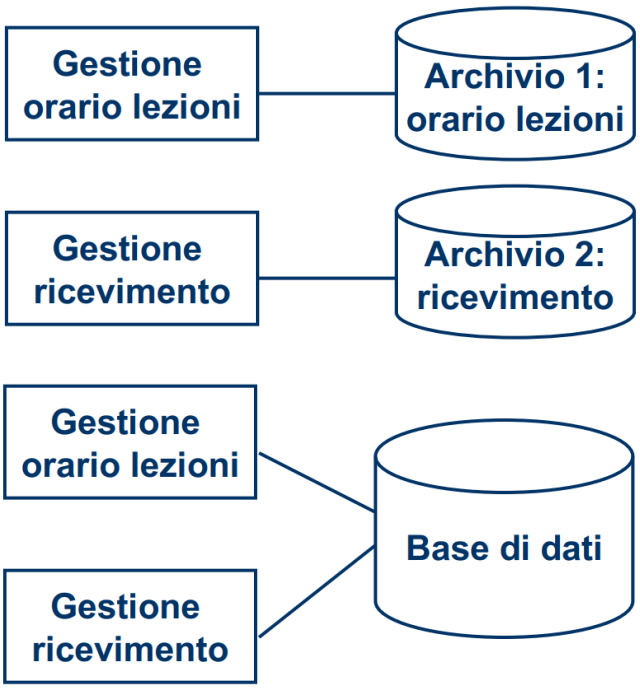
- **Privatezza**
- **Affidabilità**
- **Efficienza**
- **Efficacia**

1.7 Caratteristiche delle basi di dati

Le basi di dato sono:

- **Grandi**: dimensioni (molto) maggiori della memoria centrale dei sistemi di calcolo utilizzati. Il limite deve essere solo quello fisico dei dispositivi
- **Persistenti**: hanno un tempo di vita indipendente dalle singole esecuzioni dei programmi che le utilizzano
- **Condivise**: Ogni organizzazione (specie se grande) è divisa in settori o comunque svolge diverse attività. Ciascun settore/attività ha un (sotto)sistema informativo (non necessariamente disgiunto)
- I possibili problemi sono la **ridondanza**, cioè la ripetizione di informazioni, e **l'incoerenza**, cioè le varie versioni della basi di dati potrebbero non coincidere

1.8 Archivi e basi di dati



1.9 Condivisione delle basi di dati

Una base di dati è una risorsa **integrata e condivisa** fra applicazioni. Diventa quindi possibile svolgere attività diverse su dati condivisi: sono necessari quindi **meccanismi di autorizzazione**. Diventano anche possibili accessi di più utenti ai dati condivisi: sono quindi necessarie politiche e meccanismi di controllo della **concorrenza**.

1.10 Caratteristiche dei DBMS

I DBMS garantiscono **privatezza**; cioè si possono definire meccanismi di autorizzazione: l'utente A è, per esempio, autorizzato a leggere tutti i dati e a modificare quelli sul ricevimento; tuttavia l'utente B è magari autorizzato a leggere i dati di X e a modificare i dati di Y.
I DBMS garantiscono inoltre **l'affidabilità** dei dati, cioè la resistenza a malfunzionamenti hardware e software. Per implementare questa caratteristica è quindi fondamentale **gestire le transazioni**.

1.11 Transazioni

Le **transazioni** sono l'insieme di operazioni da considerare indivisibili ("**atomiche**"), corrette anche in presenza di **concorrenza** e con effetti **definitivi**. La sequenza di operazioni sulla base di dati viene eseguita per intero o per niente: per esempio, il trasferimento di fondi da un conto A ad un conto B avviene o tramite il prelevamento da A e il versamento su B o niente affatto. L'effetto di transazioni concorrenti deve essere **coerente** (ad esempio "equivalente" all'esecuzione separata delle richieste concorrenti): per esempio, se due assegni emessi sullo stesso conto corrente vengono incassati **contemporaneamente**, si deve evitare di trascurarne uno. La conclusione positiva di

una transazione corrisponde ad un impegno (in inglese **commit**) a mantenere traccia del risultato in modo definitivo, anche in presenza di guasti e di esecuzione concorrente.

1.12 Caratteristiche dei DBMS

I DBMS devono essere **efficienti**, cioè devono cercare di utilizzare al meglio le risorse di spazio di memoria (principale e secondaria) e di tempo (di esecuzione e di risposta). I DBMS, con tante funzioni, rischiano l'inefficienza e per questo ci sono grandi investimenti e competizione. L'efficienza è anche il risultato della qualità delle applicazioni.
I DBMS devono essere **efficaci**, cioè devono cercare di rendere produttive le attività dei loro utilizzatori, offrendo funzionalità articolare, potenti e flessibili.

1.13 Descrizioni dei dati nei DBMS

Descrizioni e rappresentazione dei dati a livelli diversi permettono **l'indipendenza dei dati** dalla rappresentazione fisica: i programmi fanno riferimento alla struttura a livello più altro, e le rappresentazione sottostanti possono essere modificate senza necessità di modificare i programmi. Possiamo precisare questo concetto attraverso il concetto di **modello dei dati**

1.13.1 Modello dei dati

Il **modello dei dati** è l'insieme di costrutti utilizzati per organizzare i dati di interesse e descriverne la dinamica. Esso ha una componente fondamentale: i **meccanismi di strutturazione** (o **costruttori di tipo**): come nei linguaggi di programmazione esistono meccanismi che permettono di definire nuovi tipi, così ogni modello dei dati prevede alcuni costruttori. Ad esempio, il **modello relazionale** prevede il costruttore **relazione**, che permette di definire insiemi di record omogenei.

1.14 Schema e istanza della base di dati

Insegnamento	Docente	Aula	Ora
Analisi matem. I	Luigi Neri	N1	8:00
Basi di dati	Piero Rossi	N2	9:45
Chimica	Nicola Mori	N1	9:45
Fisica I	Mario Bruni	N1	11:45
Fisica II	Mario Bruni	N3	9:45
Sistemi inform.	Piero Rossi	N3	8:00

In ogni base di dati esistono:

- Lo **schema**, sostanzialmente invariante nel tempo, che ne descriva la struttura e il significato (aspetto intensionale; es. le intestazioni delle tabelle). Lo schema quindi costituisce l'aspetto **intensionale**, ovvero la descrizione "astratta" delle proprietà, ed è invariante nel tempo

- **L'istanza**, i valori attuali, che possono cambiare anche molto rapidamente (aspetto estensionale; es. il "corpo" di ciascuna tabella). L'istanza quindi costituisce

invece l'aspetto **estensionale** "concreto", che varia nel tempo al variare della situazione di ciò che stiamo descrivendo

1.15 Modello dei dati logico

I modelli dei dati logici sono adottati nei DBMS esistenti per l'organizzazione dei dati:

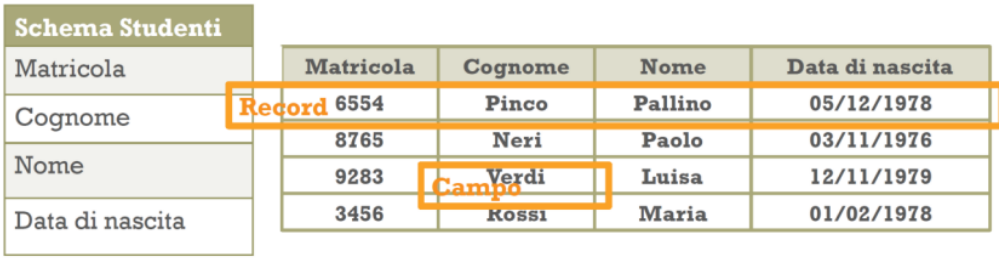
- utilizzati dai programmi
- indipendenti dalle strutture fisiche

Esempi di essi sono il modello **relazionale**, reticolare, gerarchico, a oggetti, XML, ...

1.15.1 Modello relazionale

Nel modello relazionale i dati vengono strutturati in tabelle, in particolare un DBMS relazionale può essere pensato come un insieme di tabelle. Ogni tabella mantiene informazioni di tipo omogeneo. Diverse tabelle sono collegate (in relazione) fra loro grazie alla presenza di un **campo comune**, che permette di mettere in relazione i dati delle due tabelle. Nel modello relazionale:

- **Lo schema** è la descrizione della struttura delle tabelle (stabile nel tempo)
- **L'istanza** sono i valori dei campi della tabella



1.16 Modello dei dati concettuale

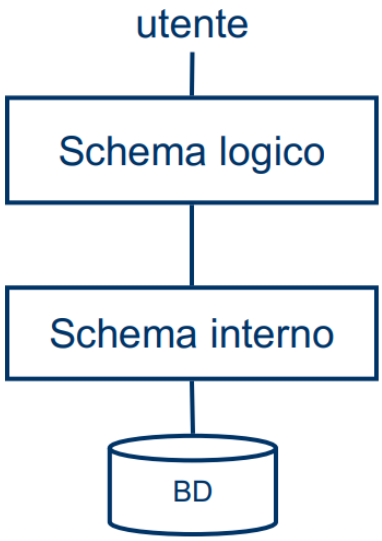
I modelli dei dati concettuali permettono di rappresentare i dati in modo indipendente da ogni sistema.

- cercando di descrivere i concetti del mondo reale
- sono utilizzati nelle fasi preliminare di progettazione (quali informazioni sono utili? Come sono collegate?)

Il più diffuso è il modello **Entity-Relationship**

1.17 Architettura di un DBMS

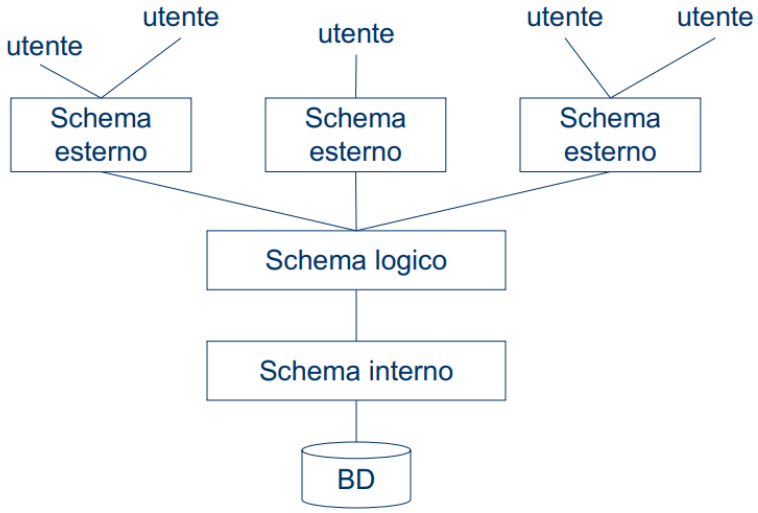
Possiamo strutturare l'architettura di un DBMS in maniera **semplificata** nel seguente modo:



Nella quale:

- **Schema logico**: descrizione della base di dati nel modello logico (ad esempio, la struttura della tabella)
- **Schema interno (o fisico)**: rappresentazione dello schema logico per mezzo di strutture di memorizzazione (file ad esempio, ma anche record con puntatori, ordinati in un certo modo)

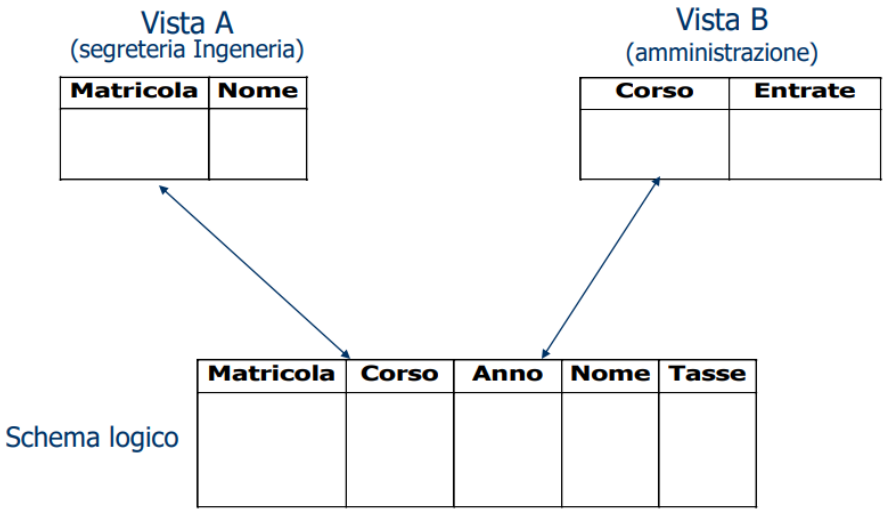
Una delle architetture standard di riferimento per i DBMS è **L'architettura ANSI/SPARC a tre livelli**:



Nella quale:

- **Schema logico**; descrizione dell'intera base di dati nel modello logico "principale" del 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)



1.18 Indipendenza dei dati

Il livello logico è indipendente da quello fisico: una tabella è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica (che può anche cambiare nel tempo). In questo corso vedremo solo il livello logico e non quello fisico. La conseguenza della articolazioni in livelli è che l'accesso avviene solo tramite il **livello esterno** (che può coincidere con il livello logico). Vi sono due forme di indipendenza:

- **Indipendenza fisica**
- **Indipendenza logica**

1.18.1 Indipendenza fisica

Il livello logico e quello esterno sono indipendenti da quello fisico. Una relazione è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica. La realizzazione fisica può cambiare senza che debbano essere modificati i programmi che usano la base di dati.

1.18.2 Indipendenza logica

Il livello esterno è indipendente da quello logico. Aggiunte o modifiche alle viste non richiedono modifiche al livello logico. Le modifiche allo schema logico che lasciano inalterato lo schema esterno sono quindi **trasparenti**.

1.19 Linguaggi per basi di dati

Vi sono due tipi di linguaggi per le basi di dati:

- **Data Definition Languages**(DDL): Servono a definire degli schemi logici, fisici e delle autorizzazioni di accesso
- **Data Manipulation Languages**(DML): Servono per effettuare l'interrogazione e l'aggiornamento delle basi di dati

Alcuni linguaggi come SQL (Structured Query Language) hanno funzioni di entrambe le categorie.

1.20 Personaggi e interpreti

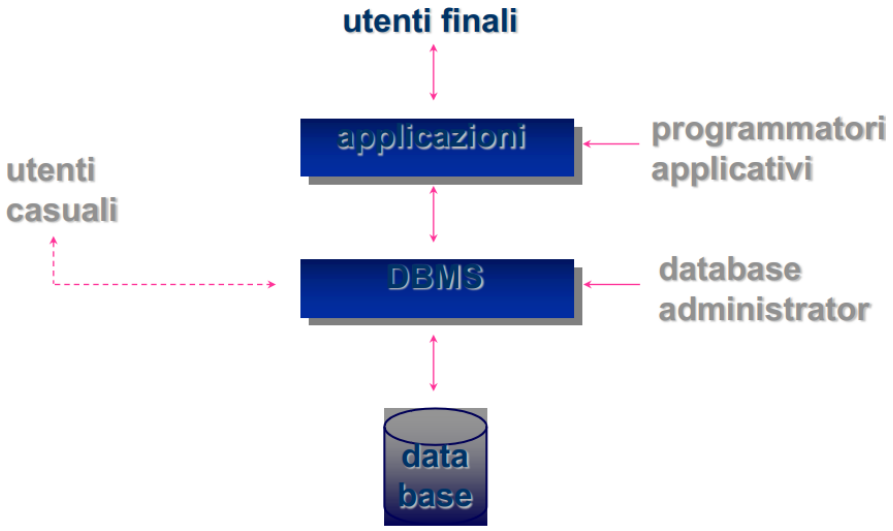
Si possono identificare diverse figure durante l'intero ciclo di vita di una base di dati:

- **Progettisti** e realizzatori di **DBMS**
- **Progettisti della base di dati** e amministratori della base di dati (**DBA**)
- **Progettisti** e programatori di **Applicazioni**
- **Utenti**
  - Utenti **finali**(terminalisti): eseguono applicazioni predefinite(**transazioni**)
  - Utenti **casuali**: eseguono operazioni non previste a priori, usando linguaggi interattivi

1.20.1 Database administrator (DBA)

Un **database administrator**(DBA) è una persona o gruppo di persone responsabile del controllo centralizzato e della gestione del sistema, delle prestazioni, dell'affidabilità e delle autorizzazioni. Le funzioni del DBA includono quelle di progettazione, anche se in progetti complessi ci possono essere distinzioni.

1.20.2 Utenti del DBMS



1.21 Vantaggi e svantaggi dei DBMS

Vediamo quali sono i **vantaggi** di avere un DBMS:

- Permettono di considerare i dati come risorsa comune di un'organizzazione, a disposizione di molteplici applicazioni e utenti
- Offrono un modello della parte di mondo di interesse che è unificato e preciso, utilizzabile in applicazioni attuali e future
- Offrono un controllo centralizzato dei dati, riducendo ridondanze e inconsistenze
- Permettono l'indipendenza dei dati: favoriscono lo sviluppo di applicazioni flessibili e facilmente modificabili

Vediamo quali sono invece gli **svantaggi** di avere un DBMS:

- Un DBMS è costoso, complesso e ha specifici requisiti in termini di software e hardware
- Diventa difficile separare, tra tutti i servizi offerti da un DBMS, quelli effettivamente usati da quelli inutili
- Sono inadatti alla gestione di applicazioni con pochi utenti (NB: Dipende dai costi operativi)

1.22 Caratteristiche dell'approccio con basi di dati

L'approccio alla **rappresentazione dei dati tramite basi di dati** ha le seguenti caratteristiche:

- **Astrazione dei dati**: si usa un **modello dati** per nascondere dettagli e presentare all'utente una *vista concettuale* del database
- **Supporto di viste multiple dei dati**: Ogni utente può usare una vista (view) differente del database, contente solo i dati di interesse per quell'utente

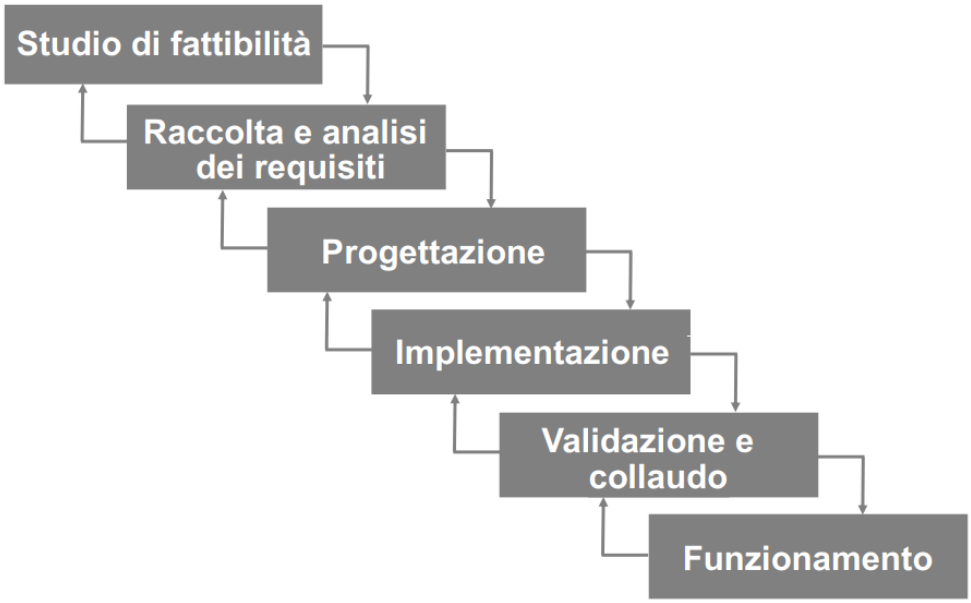
2 Progettazione di unaa base di dati

La **progettazione di basi di dati** è una delle attività del processo di sviluppo dei sistemi informativi e va quindi inquadrata in un contesto più generale: **il ciclo di vita dei sistemi informativi**.

2.1 Ciclo di vita dei sistemi informativi

Il ciclo di vita dei sistemi informativi è **l'insieme e sequenzializzazione delle attività svolte da analisti, progettisti, utenti nello sviluppo e nell'uso dei sistemi informativi**. È un attività iterativa, quindi è un "ciclo".

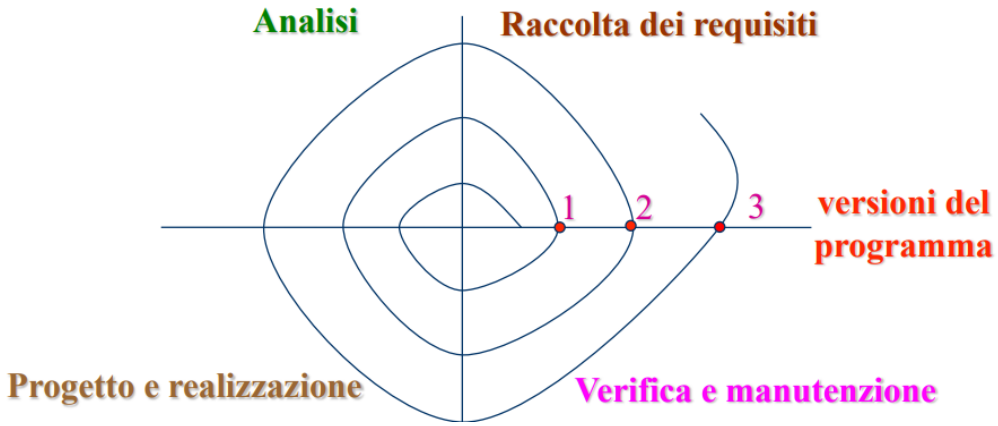
Le fasi del ciclo di vita sono:



Diamone una definizione:

- **Studio di fattibilità**: definizione dei costi e delle priorità
- **Raccolta e analisi dei requisiti**: studio delle proprietà del sistema
- **Progettazione**: di dati e funzioni
- **Validazione e collaudo**: sperimentazione
- **Funzionamento**: il sistema diventa operativo

Possiamo anche rappresentare questo ciclo di vita come una spirale:



2.1.1 Progettazione

La progettazione dei dati individua l'organizzazione e la struttura della base di dati. La progettazione delle applicazioni schematizza le operazioni sui dati e progetta il software applicativo. I dati hanno **un ruolo centrale** e grazie alla progettazione:

- i dati sono **più stabili**
- Si progetta prima la base di dati e poi le applicazioni

2.1.2 Metodologia di progettazione

Per garantire prodotti di buona qualità è opportuno seguire **una metodologia di progetto**; cioè un'articolazione in fasi/passi di guida ad una attività di progettazione. Serve una metodologia di progettazione (insieme di strumenti) che:

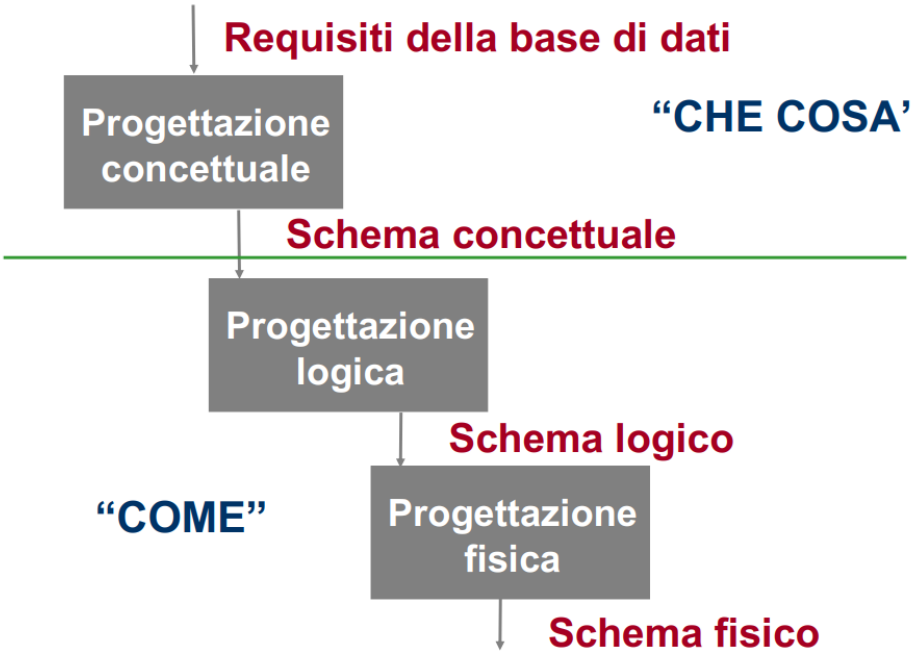
- permetta di **suddividere** la progettazione in fasi successive indipendenti
- fornisca **strategie** da seguire e criteri di scelta in caso di alternative
- fornisca **modelli di riferimento (linguaggi)** per descrivere la realtà che stiamo progettando

e che garantisca:

- **Generalità** rispetto ai problemi da affrontare
- **Qualità** in termini di correttezza, completezza ed efficienza
- **Facilità d'uso**

Una metodologia di progettazione di basi di dati si basa su un principio semplice ma efficace: **la separazione netta tra decisioni relative a**:

- **Come rappresentare**
- **Come farlo**



La metodologia introdotta prevede 3 fasi:

- Progettazione **concettuale**
- Progettazione **logica**
- Progettazione **fisica**

Ognuna delle fasi si basa su un **modello**, che permette di generare una rappresentazione formale (schema) della base di dati ad un dato livello di astrazione (concettuale, logico e fisico):

- Schema concettuale
- Schema logico
- Schema fisico

2.1.3 Fase di progettazione concettuale

La fase diu progettazione concettuale traduce i requisiti del sistema informatico in una descrizione **formalizzata e integrata** delle esigenze aziendali, espressa in modo **indipendente** dalle scelte implementative (DBMS, SW e HW). Essa è:

- **Formale**: la descrizione deve essere espressa con un linguaggio non ambiguo e capace di descrivere in modo soddisfacente il sistema analizzato
- **Integrata**: la descrizione deve essere in grado di descrivere nella globalità l'ambiente analizzato
- **Indipendente dall'ambiente tecnologico**: la descrizione deve concentrarsi sui dati e sulle loro relazioni e non sulle scelte implementative

2.1.4 Fase di progettazione logica

La progettazione logica consiste nella traduzione dello schema concettuale nel modello dei dati del DBMS. Il risultato è uno schema logico, espresso nel DDL del DBMS. In questa fase si considerano anche aspetti legati ai vincoli e all'efficienza. La progettazione logica si articola in due sotto-fasi:

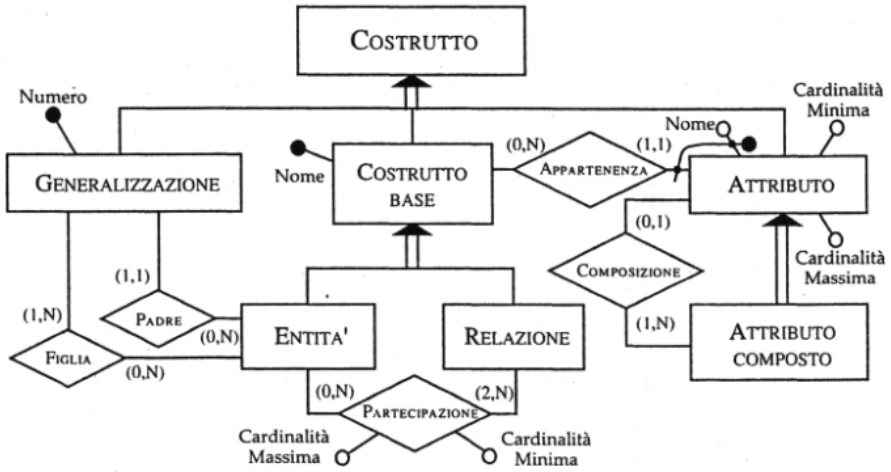
- **Ristrutturazione dello schema concettuale**
- **Traduzione verso il modello logico**

2.1.5 Progettazione fisica

La progettazione fisica completa lo schema logico ottenuto con le specifiche proprie dell'hw/sw scelto. Il risultato è lo **schema fisico** che descrive le strutture di memorizzazione ed accesso ai dati.

2.1.6 Vantaggi della progettazione concettuale

La progettazione concettuale permette una descrizione dei dati indipendete degli aspetti tecnologici con un livello di astrazione intermedio fra utente e sistema. **Prevale l'aspetto intensionale**. Inoltre, essa utilizza una rappresentazione prevalentemente **grafica**, che migliora la comunicazione tra i progettisti, gli utenti e tutte le persone coinvolte nella realizzazione dell'applicazione. È inoltre utile per la **documentazione**.

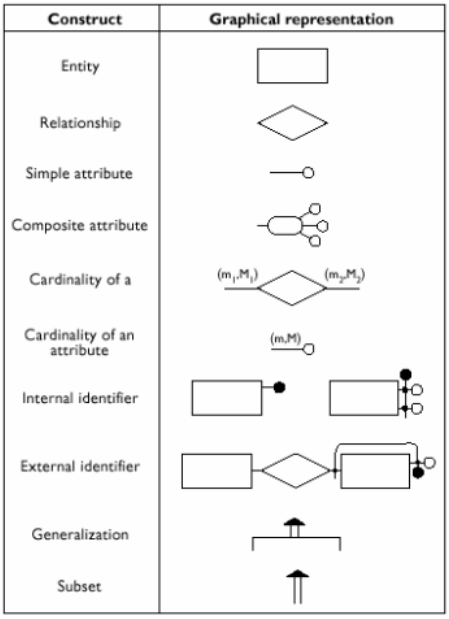


3 Modello Entità-Relazione (E-R)

Il modello entità-relazione (E-R) è un linguaggio grafico semi-formale per la rappresentazione di schemi concettuali. Il modello E-R si è ormai affermato come uno standard nelle metodologie di progetto e nei sistemi SW di ausilio alla progettazione. Ne esistono molte versione più o meno diverse l'una dall'altra.

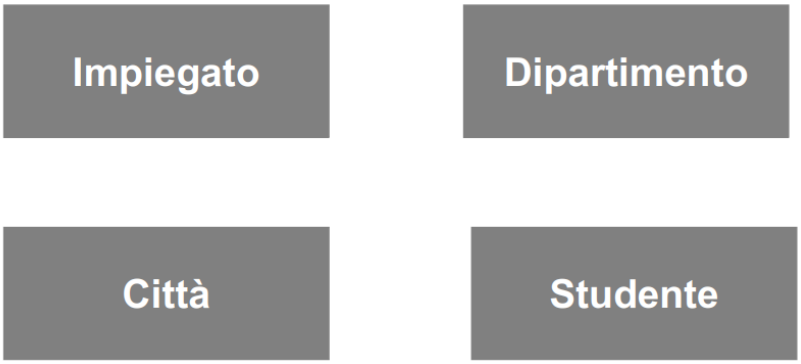
3.1 Introduzione ai costrutti

- **entità**
- **relazione**
- **attributo semplice**
- **attributo composto**
- **cardinalità**
- **cardinalità di un attributo**
- **identificatore interno**
- **identificatore esterno**
- **generalizzazione**
- **sottoinsieme**



3.2 Entità

Un'entità è una classe di oggetti (fatti, persone, cose) della applicazione di interesse con proprietà comuni e con esistenza **autonoma** e della quale si vogliono registrare fatti specifici. Per esempio, un'entità potrebbe essere un impiegato, un dipartimento, una città, un conto corrente, un'università, uno studente, ... Graficamente le entità si rappresentano nel seguente modo:



Ogni entità ha un nome che la identifica univocamente nello schema, il quale deve essere:

- **Espressivo**
- **Al singolare**

A livello **estensionale** un'entità è costituita da un insieme di oggetti, che sono chiamati le sue **istanze**. Ciò significa che, se in uno schema S è definita una entità E, in ogni istanza I dello schema S, alla entità E è associato un insieme di oggetti

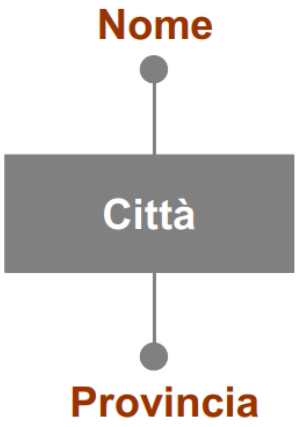
$$istanze(I, E) = \{e_1, e_2, e_3, \dots\}$$

che viene detto anche *l'estensione* di E nella istanza I dello schema S. Una istanza di entità non è un valore che identifica un oggetto, ma è **l'oggetto stesso**. Quindi ogni istanza di entità è **oggetto della classe che l'entità rappresenta**. Nello schema concettuale rappresentiamo le **entità**, non le singole istanze ("astrazione"). Quindi:

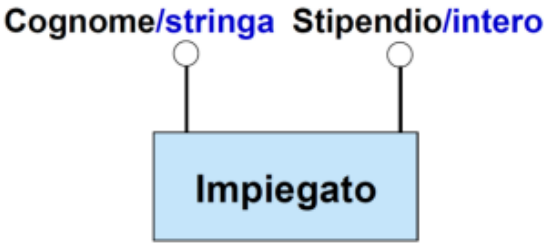
- **Conoscenza astratta**: entità
- **Conoscenza concreta**: istanza di entità

3.3 Attributi

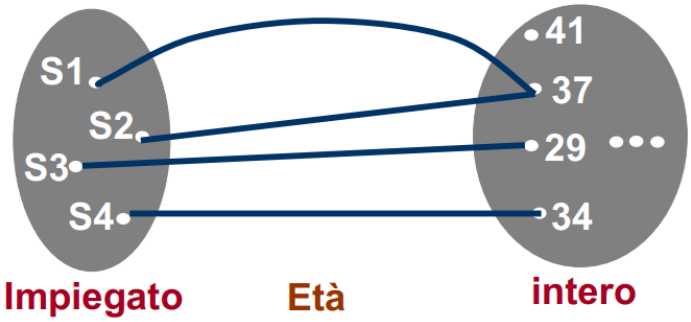
Un attributo di entità è una proprietà locale di un'entità, di interesse ai fini dell'applicazione. Un attributo associa ad ogni istanza di entità un valore appartenente ad un insieme detto **dominio dell'attributo** (tipicamente interi, caratteri, stringhe, ecc...). Si definisce quindi un attributo per l'entità E quando si vuole rappresentare una proprietà **locale** delle istanze dell'entità E. Una proprietà di un oggetto si dice locale quando in ogni istanza dello schema il valore di tale proprietà dipende **solamente** dall'oggetto stesso e **non ha alcun rapporto** con altri elementi dell'istanza dello schema. Nel modello E-R, gli attributi si rappresentano nella seguente maniera:



Ogni attributo di entità ha un nome che lo identifica in modo univoco nell'ambito dell'entità, ed è rappresentato da un cerchio collegato alla entità a cui appartiene. Ogni attributo è definito su un dominio di valori. Un attributo associa ad ogni istanza di entità o associazione un valore nel corrispondente dominio. I domini solitamente non vengono specificati nell'E-R ma nella documentazione associata. Se li si vuole indicare, la notazione è la seguente:



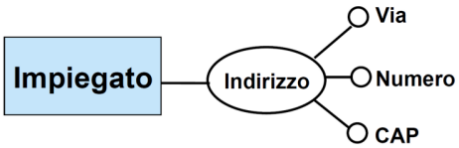
Quindi, ogni attributo può essere pensato come una vera e propria **funzione** che associa una determinata istanza di un'entità ad un particolare valore per quell'attributo; di conseguenza **valgono tutte le definizioni e tutte le proprietà delle funzioni**



Inoltre, un'attributo è una **funzione totale**, quindi ogni istanza di un'entità deve avere un valore associato per ogni suo attributo.

3.4 Attributi composti

Gli attributi composti si ottengono raggruppando attributi di una medesima entità o relazione che presentano affinità nel loro significato o uso:



3.5 Relazioni

Una relazione è un fatto che descrive un'azione o una situazione che **stabilisce legami logici** tra istanze di entità (associa, mette in relazione) nella **realtà che stiamo considerando**. **I legami possono essere fra più di due entità**. Il numero di entità coinvolte in una relazione determina il suo **grado**. Nel modello E-R una relazione si indica nel seguente modo:

Ogni relazione ha un nome che la identifica univocamente nello schema ed esso deve essere:

- **Espressivo**
- **Singolare** e usare **sostantivi invece che verbi**

A livello **estensionale** una relazione R tra le entità E ed F è costituita da un insieme di coppie (x, y), tali che x è una istanza di E ed y è un istanza di F. Ogni coppia è detta **istanza** della relazione R. cIò significa che, se in uno schema S è definita una relazione R sulle entità E ed F, in ogni istanza I dello schema S, alla relazione R è associato un insieme di coppie (denotato da istanze (I, R))

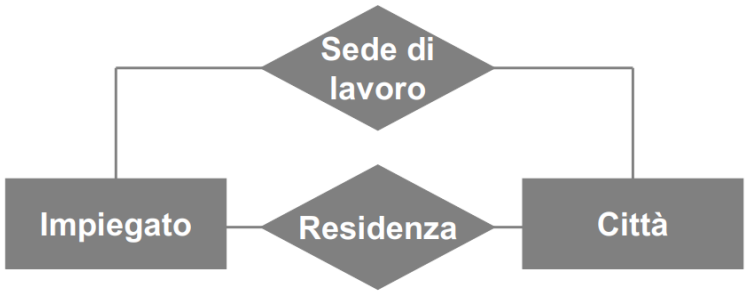
$$\{(x_1, y_1)(x_2, y_2), (x_3, y_3), \dots\}$$

che viene detto anche *l'estensione* di R nella istanza I dello schema S. In altre parole, una relazione nel modello E-R è, dal punto di vista della semantica, una relazione matematica. In ogni istanza I dello schema S si ha:

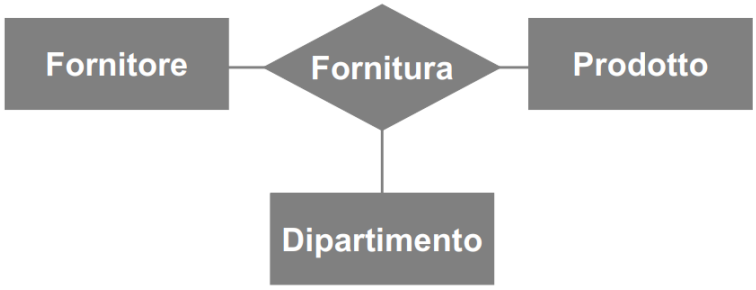
$$istanze(I, R) \subseteq istanze(I, E) \times istanze(I, F)$$

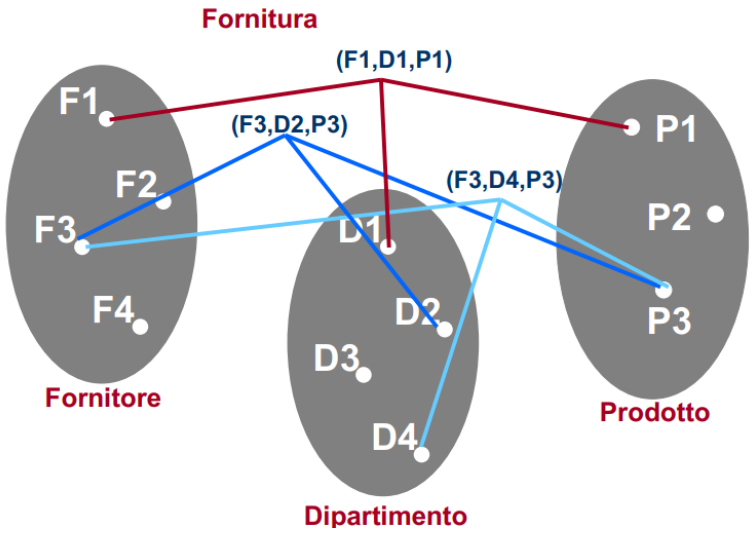
In particolare si dice **istanza di un'associazione** la combinazione (aggregazione) di istanze di entità che prendono parte alla associazione. Dalla semantica delle relazioni segue immediatamente che **non** possono esistere due istanze della stessa relazione che coinvolgono le stesse istanze di entità.

Tuttavia **due entità possono essere coinvolte in più associazioni**:



Inoltre le relazioni possono coinvolgere più di due entità:





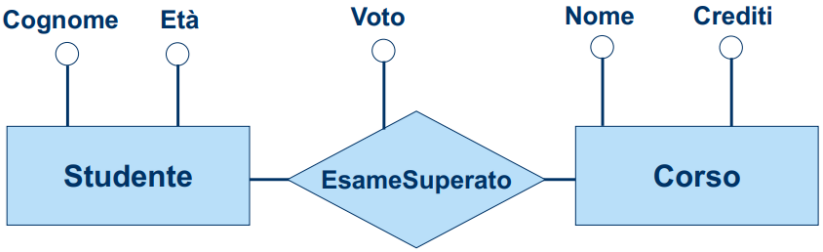
Se si parla quindi di una relazione **n-aria**, a livello **estensionale** (ovvero in ogni istanza I dello schema S) una relazione R tra le entità $E_1, E_2, \ldots, E_n$ è costituita dun insieme di n-uple (o tuple) $(x_1, x_2, \ldots, x_n)$, tali che x_1 è un istanza di E_1 in I, x_2 è un istanza di E_2 in I, $\ldots x_N$ è un istanza di E_n in I. Ogni n-upla è detta **istanza** della relazione R nella istanza I dello schema S. Quindi, in ogni istanza I dello schema si ha:

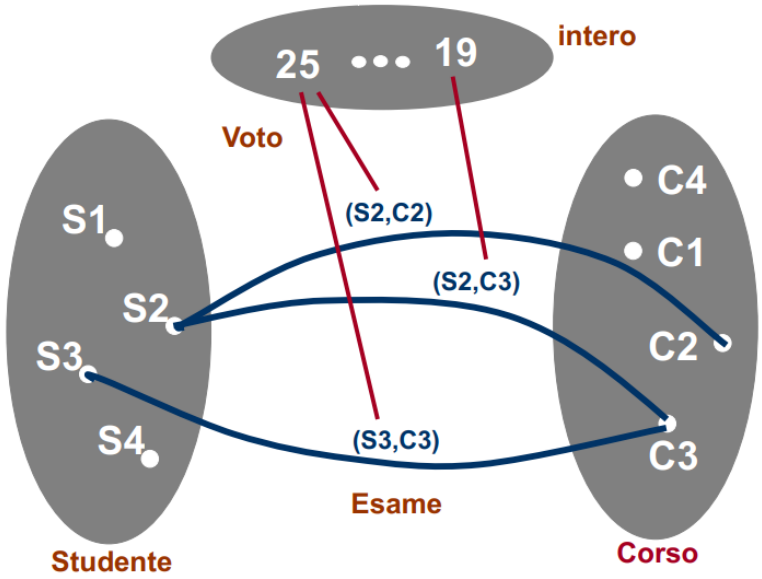
$$istanze(I, R) \subseteq istanze(I, E_1) \times \cdots \times istanze(I, E_n)$$

3.6 Attributi di relazione

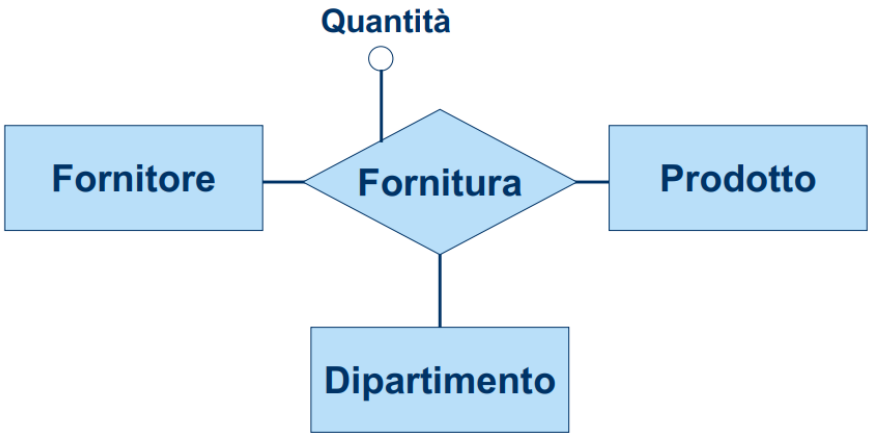
Un attributo di relazione è una proprietà locale di una relazione, di interesse ai fini dell'applicazione. Un attributo della relazione R tra le entità $E_1, E_2, \ldots, E_n$ modella

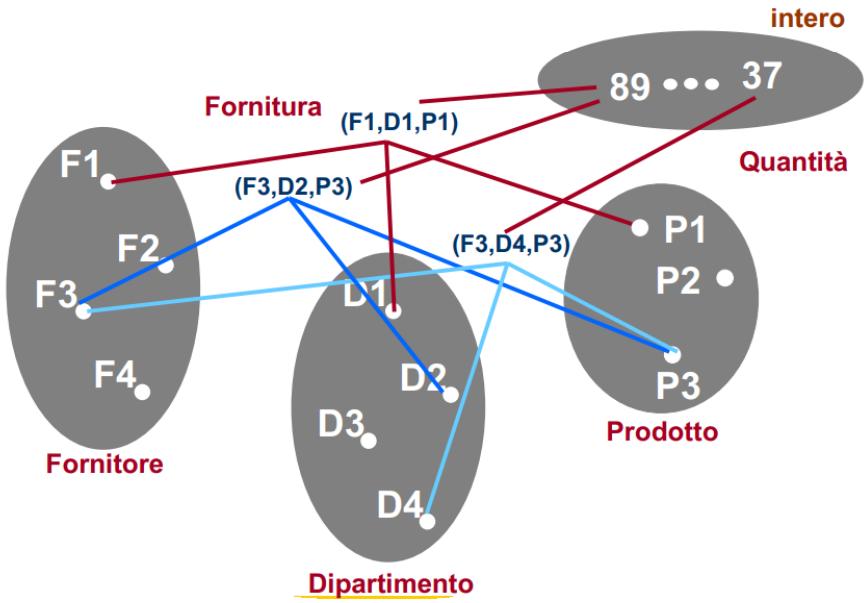
una proprietà **non** propria delle entità ma del **legame** tra le varie entità rappresentato da R. Un attributo associa ad ogni istanza di relazione un valore appartenente ad un insieme detto **dominio** dell'attributo. Ogni attributo di relazione ha un nome che lo identifica in modo univoco nell'ambito della relazione, ed è rappresentato da un cerchio collegato alla relazione a cui appartiene.





Ovviamente gli attributi possono essere presenti su associazioni n-arie:



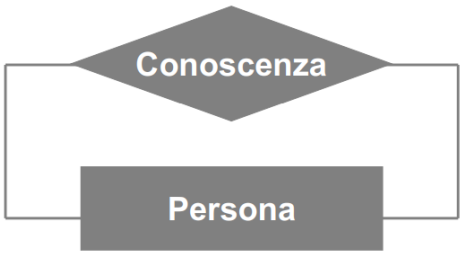


3.7 Analisi dei casi dubbi

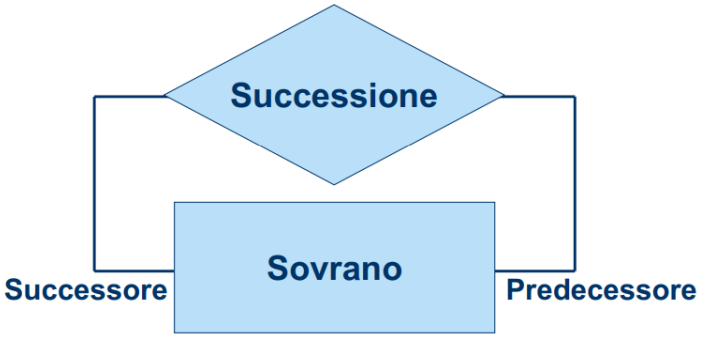
I nomi di entità e associazioni alle volte traggono in inganno: è bene quindi, nel caso si presentino situazioni poco chiare, bisogna provare a ragionare anche in termini di istanze: cosa "contiene" effettivamente questa entità/associazione?

3.8 Relazioni ricorsive

Una associazione può coinvolgere "due o più volte" la stessa entità:



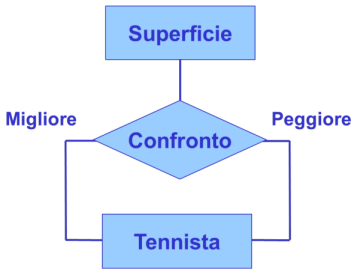
Vi è però il problema di **identificare i ruoli in questa associazione**. Per farlo, essi si scrivono di fianco ai "collegamenti" che collegano la relazione con l'entità:



Un'associazione ricorsiva può essere o meno:

- **Simmetrica**: $(a, b) \in A \Rightarrow (b, a) \in A$
- **Riflessiva**: $(a, a) \in A$
- **Transitiva**: $(a, b) \in A, (b, c) \in A \Rightarrow (a, c) \in A$

Una associazione ricorsiva può essere anche n-aria:



3.9 Scelta tra entità, relazione e attributo

Un concetto verrà modellato come:

- **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
- **Attributo** di un'entità o di una relazione:
  - 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
- **Relazione**:
  - Se le sue istanze non sono concettualmente significative indipendentemente da altre istanze, cioè le sue istanze rappresentano n-uple di altre istanze
  - Se non ha senso pensare alla partecipazione delle sue istanze ad altre relazioni

Le scelte che si prendono possono cambiare durante l'analisi.

3.10 Cardinalità

La cardinalità è una coppia di valori che si associa a ogni entità che partecipa a una relazione. Specificano il numero minimo e massimo di occorrenze della relazione cui ciascuna occorrenza di una entità può partecipare. Per semplicità usiamo solo tre simboli:

- 0 e 1 per la cardinalità **minima**
  - 0 = "partecipazione **opzionale**"
  - 1 = "partecipazione **obbligatoria**"
- 1 e "N" per la **massima**; con N che non pone alcun limite

Nel modello E-R le cardinalità si rappresentano nel seguente modo:

Con riferimento alle cardinalità **massime**, abbiamo relazioni:

- **Uno a uno**: se le cardinalità massime di entrambe le entità sono uno
- **Uno a molti**: se la cardinalità massima associata ad un'entità è 1 e la cardinalità massima associata all'altra entità è N
- **Molti a molti**: se le cardinalità massime per entrambe le entità è N

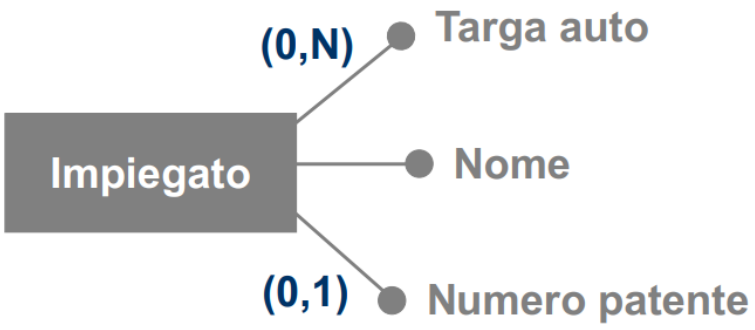
3.10.1 Vincoli di cardinalità

Un **vincolo di cardinalità** tra un'entità E e una relazione R esprime un limite minimo (**cardinalità minima**) ed un limite massimo (**cardinalità massima**) di istanze della relazione R a cui può partecipare ogni istanza dell'entità E. Serve a caratterizzare meglio il significato di una relazione.

3.10.2 Cardinalità di attributi

È possibile associare delle cardinalità anche agli attributi, con due scopi:

- Indicare **opzionalità**
- Indicare **attributi multivalore**



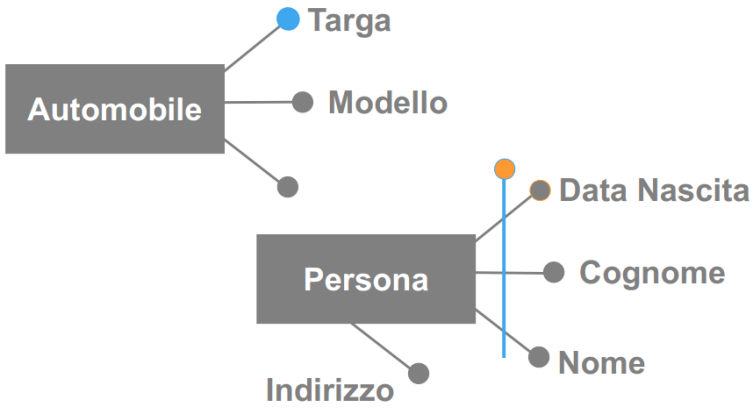
3.11 Identificatori di un'entità

L'identificatore è uno "strumento" per l'identificazione univoca delle occorrenze di un'entità. Esso è costituito da

- **Attributi dell'entità**; in questo caso è detto **identificatore interno**
- **Attributi dell'entità più entità esterne attraverso relazioni**; in questo caso è detto **identificatore esterno**

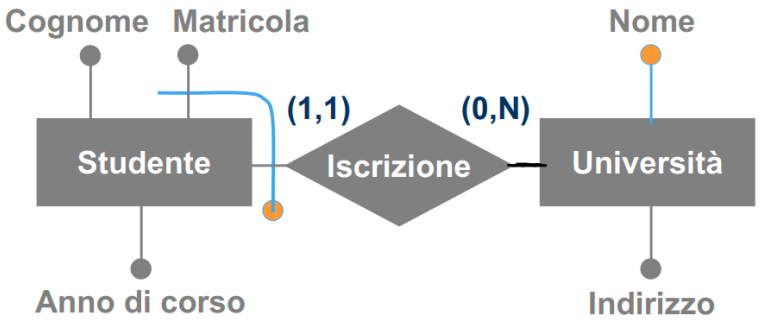
Per gli **identificatori interni** si segue la seguente notazione:

- Se l'identificatore è formato da un solo attributo, si annerisce il corrispondente pallino
- Se l'identificatore è formato da più attributi, si uniscono gli attributi con una linea che termina con pallino annerito



Per gli **identificatori esterni** si segue la seguente notazione:

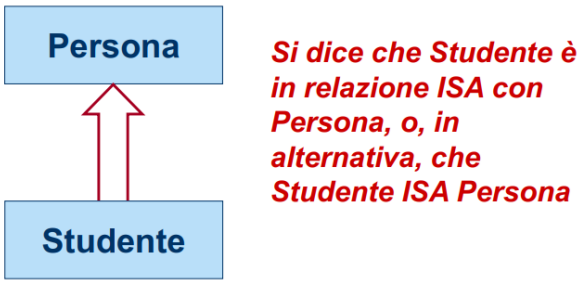
- Se l'identificatore è formato da attributi e relazioni (o meglio, ruoli), si indica unendo gli attributi ed i ruoli con una linea che termina con pallino annerito.
- Una identificazione esterna è possibile solo **attraverso una relazione a cui l'entità da identificare partecipa con cardinalità (1,1)**



Ogni entità deve possedere almeno un identificatore, ma può averne in generale più di uno.

3.12 Relazione IS-A tra entità

Fino ad ora non abbiamo detto nulla sul fatto se due entità possano o no avere istanze in comune. 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**. (Es. Studente, Studente della laurea breve). La relazione IS-A nel modello E-R si può definire tra due entità, che si dicono rispettivamente "entità padre" ed "entità figlia" (o sottoentità, cioè quella che rappresenta un sottoinsieme dell'entità padre). La relazione IS-A si rappresenta nel diagramma dello schema concettuale nel seguente modo:



3.12.1 Ereditarietà su entità

La relazione IS-A implica il **principio di ereditarietà**: ongi 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.

3.13 Generalizzazione tra entità

Finora, abbiamo considerato la relazione IS-A 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 E-R, si può specificare se due sottoentità della stessa entità padre sono disgiunte o no). Una generalizzazione può essere di due tipi:

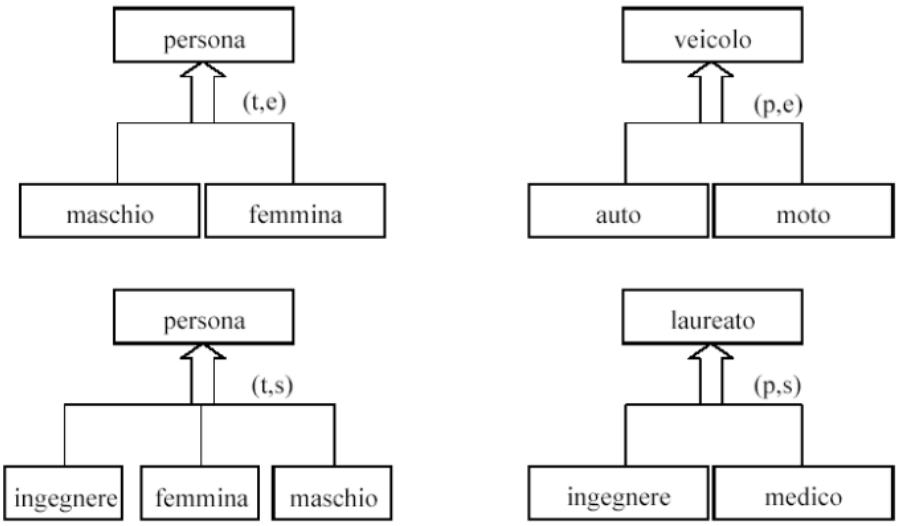
- **Completa**: l'unione delle istanze delle sottoentità è uguale all'insieme delle istanze dell'entità padre; si indica con una freccia annerita
- **Non completa**: si indica con una freccia non riempita

Alla generalizzazione si associa una coppia di lettere che indica come interpretare la generalizzazione. La prima lettera indica il tipo di generalizzazione e può assumere i valori:

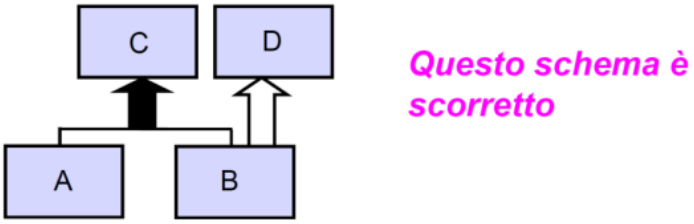
- **t** per le generalizzazione complete (totali)
- **p** per le generalizzazioni non complete (parziali)

La seconda lettera invece indica se le sottoentità si sovrappongono o meno; essa può assumere i valori:

- **e** se le sottoentità non si sovrappongono (esclusiva)
- **s** se le sottoentità si sovrappongono (sovrapposte)



Nel modello E-R vige la regola che una entità può avere **al massimo una** entità padre. In altre parole, il modello E-R **non ammette ereditarietà multipla**. Ovviamente, quando si contano i padri di una sottoentità si tiene conto anche di eventuali relazioni **IS-A** che essa ha con altre entità.



3.13.1 Generalizzazioni ed ereditarietà

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'entita figlia può avere ovviamente ulteriori proprietà.

3.13.2 Diverse generalizzazioni della stessa classe

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 delle entità padre.

3.13.3 Differenza tra due IS-A e una generalizzazione

Ci si potrebbe chiedere se non si possa utilizzare due IS-A per rappresentare una generalizzazione. In realtà, le due (o più) sottoclassi che derivano da una generalizzazione derivano da uno stesso criterio di classificazione delle istanze della superclasse. Le due sottoentità di due IS-A differenti invece sono indipendenti, nel senso che il loro significato non deriva dallo stesso criterio di classificazione delle istanze della entità padre.

3.13.4 Altre proprietà

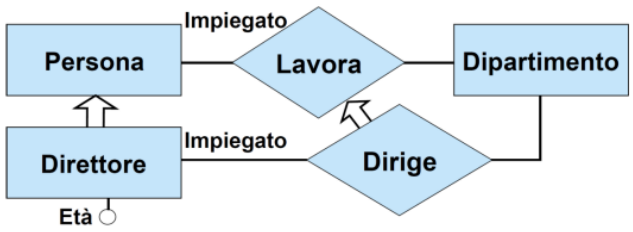
Vi sono altre proprietà della generalizzazione:

- 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**

3.14 Relazione IS-A fra relazioni

Nel caso di una relazione IS-A fra relazioni la semantica non cambia rispetto al caso della relazione IS-A tra entità: se in uno schema S è definita la relazione IS-A tra R e Q (R IS-A Q, dove R e Q sono due relazioni con lo stesso grado e gli stessi ruoli), allora in ogni istanza I dello schema S, si ha che

$$istanze(I, R) \subseteq istanze(I, Q)$$



Il direttore di un dipartimento deve lavorare in quel dipartimento, ed ha proprietà aggiuntive (ad esempio l'età) rispetto agli impiegati

3.15 Vincoli non esprimibili nel diagramma E-R

Gli schemi E-R permettono di cogliere la maggior parte delle interrelazioni tra i dati del dominio d'interesse. Tuttavia alcune interrelazioni non possono essere colte direttamente da uno schema E-R. Talli interrelazioni vanno in ogni caso tenute presenti attraverso delle asserzioni aggiuntive dette **vincoli esterni al diagramma**, o semplicemente, **vincoli esterni**. Come rappresentiamo tali vincoli?

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

3.16 Documentazione associata agli schemi E-R

Oltre al diagramma E-R, lo schema concettuale è descritto dal cosiddetto **dizionario dei dati**: esso è costituito dalle tabelle di

- Entità
- Relazioni
- Attributi
- Vincoli esterni

Dizionario dei dati: entità

Entità	Descrizione	Attributi	Identificatori
Impiegato	Dipendente dell'azienda	Codice Cognome Stipendio Anzianità	{ Codice }
Progetto	Progetti aziendali	Nome Budget	{ Nome }
Dipartimento	Struttura aziendale	Nome Telefono	{ Nome, Sede }
Sede	Sede dell'azienda	Città Indirizzo (Via, CAP)	{ Città, Indirizzo }

Dizionario dei dati: relazioni

Relazione	Descrizione	Componenti	Attributi
Direzione	Direzione di un dipartimento	Impiegato, Dipartimento	
Afferenza	Afferenza ad un dipartimento	Impiegato, Dipartimento	Data
Partecipazione	Partecipazione ad un progetto	Impiegato, Progetto	
Composizione	Composizione dell'Azienda	Dipartimento, Sede	

Dizionario dei dati: attributi

Attributo	Entità/Relazione	Dominio	Descrizione
Codice	Impiegato	Intero	Codice identificativo di impiegati
Cognome	Impiegato	Stringa	Cognome di impiegato
Stipendio	Impiegato	Reale	Stipendio di impiegato
Nome	Progetto	Stringa	Nome del progetto
...	...	...	...

Dizionario dei dati: vincoli esterni

Vincoli di integrità esterni
(1) Il direttore di un dipartimento deve afferire a tale dipartimento
(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 che non afferisce ad alcun dipartimento non deve partecipare ad alcun progetto

In particolare, il dizionario dei vincoli esterni può avere, a sua volta, altri vincoli esterni espressi in un altro dizionario.

4 Progettazione concettuale

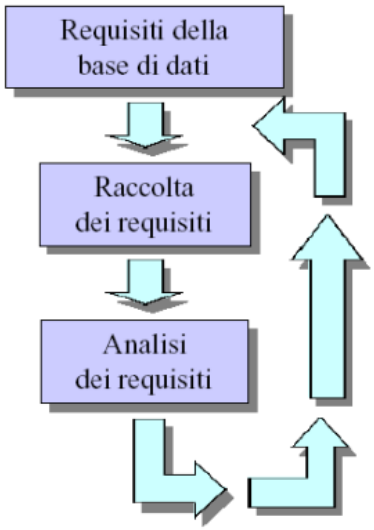
L'analisi dei requisiti e la progettazione concettuale ("analisi dei dati") comprende attività interconnesse di:

- Acquisizione dei requisiti
- Analisi dei requisiti
- Costruzione dello schema concettuale
- Costruzione del glossario

Per i requisti, possibili fonti sono:

- **Utenti**, attraverso:
  - Interviste
  - Documentazione apposita
- **Documentazione esistente**
  - Normative (leggi, regolamenti di settore)
  - Regolamenti interni, procedure aziendali
  - Realizzazioni preesistenti
- **Modulistica**

Il reperimento dei requisiti è un 'attività difficile e non standardizzabile. L'attività di anlisi iniza con i primi requisiti raccolti e spesso indirizza verso altre acquisizioni.



4.1 Requisiti

4.1.1 Acquisizione per interviste

Utenti diversi possono fornire informazioni diverse. Utenti a livello più alto hanno spesso una visione più ampia ma meno dettagliata. Le interviste portano spesso ad una acquisizione dei requisiti "per raffinamenti successivi".

4.1.2 Interazione con gli utenti

Spunti per ottenere requisiti tramite l'interazione con gli utenti possono essere:

- Effettuare spesso verifiche di comprensione e coerenza
- Verificare anche per mezzo di esempi (generali e relativi ai casi limite)
- Richiedere definizioni e classificazioni
- Far evidenziare gli aspetti essenziali rispetto a quelli marginali

4.1.3 Documentazione descrittiva

Le regole generali sono:

- Scegliere il corretto livello di astrazione
- Standardizzare la struttura delle frasi
- Suddividere le frasi articolate
- Separare le frasi sui dati da quelle sulle funzioni

4.1.4 Organizzazione di termini e concetti

Le regole generali sono:

- Costruire un glossario dei termini
- Individuare omonimi e sinonimi e unificare i termini
- Rendere esplicito il riferimento fra termini
- Riorganizzare le frasi per concetti

L'analisi dei requisiti procede per **raffinamenti successivi**.

4.1.5 Scrittura dei requisiti

Per scrivere correttamente i requisiti serve:

- **Scegliere il livello di astrazione**: Si devono evitare termini troppo generici o troppo specifici
- **Standardizzare la struttura delle frasi**: È preferibile usare sempre lo stesso stile sintattico
- **Evitare frasi contorte**
- **Individuare sinonimi/omonimi e unificare i termini**: Usare un solo termine al posto dei sinonimi; usare termini distinti in caso di omonimi
- **Rendere esplicito il riferimento fra termini**: In assenza di un contesto di riferimento alcuni termini possono risultare ambigui. Occorre rappresentare il riferimento fra i termini
- **Costruire un glossario dei termini (facoltativo)**: per ogni termine si indica una breve descrizione, i sinonimi e gli altri termini con cui esiste un legame logico. Facilita la comprensione e la precisazione dei termini

4.2 Specifiche sulle operazione

Per scrivere le specifiche sulle operazioni occorre utilizzare la stessa terminologia utilizzata per le specifiche dei dati (glossario). Serve anche conoscere la frequenza con la quale le operazioni sono eseguite.

4.3 Dalle specifiche al modello E-R

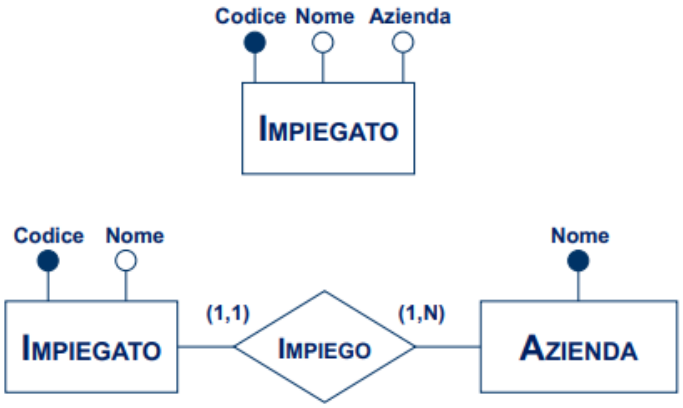
Per passare dalle specifiche al modello E-R si utilizzano le definizioni dei costrutti del modello E-R. Se un concetto proprietà significative e descrive classi di oggetti con esistenza autonoma conviene rappresentarlo con una **entità**. Se un concetto ha una struttura semplice e non possiede proprietà rilevanti associate conviene rappresentarlo come un **attributo** di un altro concetto a cui si riferisce. Se sono state individuate due o più entità e nei requisiti compare un concetto che le associa, questo concetto può essere rappresentato con una **relazione**. Se uno pià concetti risultano essere il caso particolare di un altro è opportuno rappresentarli facendo uso della **generalizzazione**.

4.4 Design patterns

I **design patterns** sono soluzioni progettuali a problemi comuni. Sono largamente usati nell'ingegneria del software. I design pattern per le basi di dati utilizzano il concetto di **reificazione**, cioè il procedimento di creazione di un modello dati su un concetto astratto predefinito.

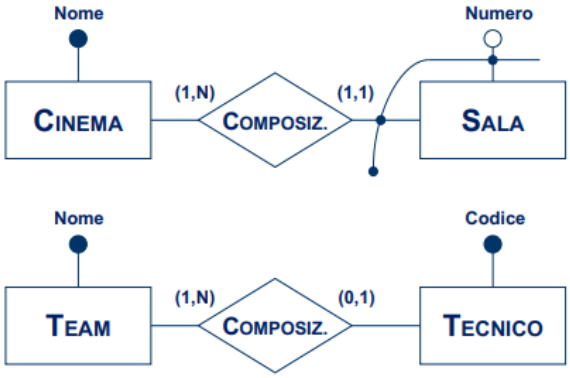
4.4.1 Reificazione di attributo di entità

Questo design pattern estrae un attributo e lo rende entità poiché, nel contesto applicativo, esso ha dignità di entità



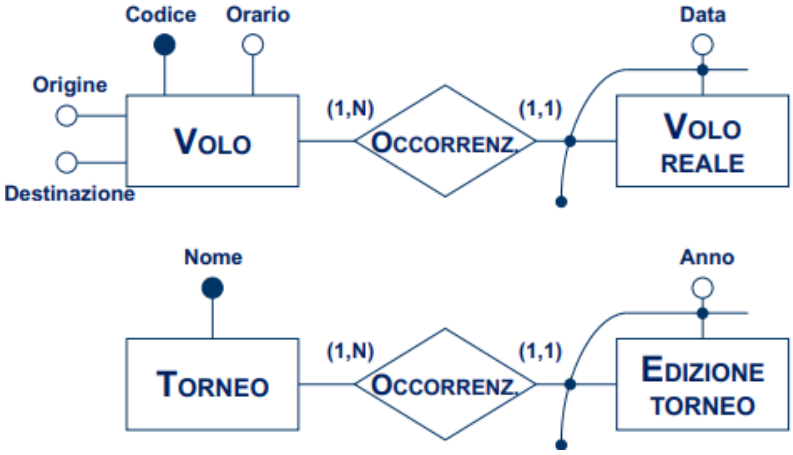
4.4.2 Part-of

A volte un'entità può essere legata a un'altra entità creando una relazione di tipo (1, N). Gli esempi mostrano come il concetto di part-of possa essere di **dipendenza**.

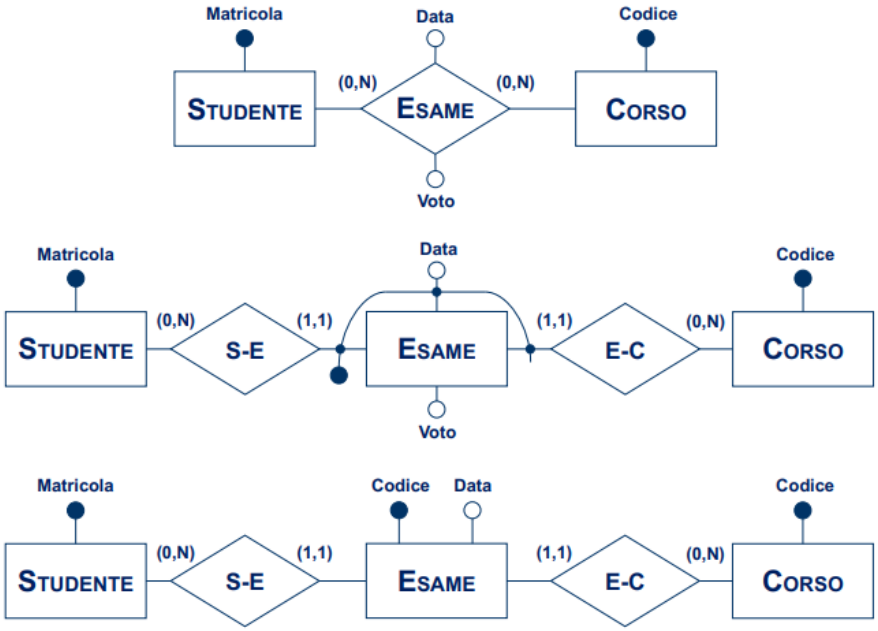


4.4.3 Instance-of

A volte si viene a creare la necessità di creare un'entità astratta che prenda concretezza in un'entità istanza.



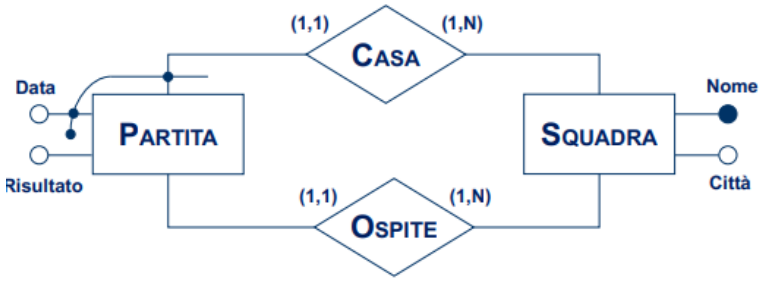
4.4.4 Reificazione di relazione binaria



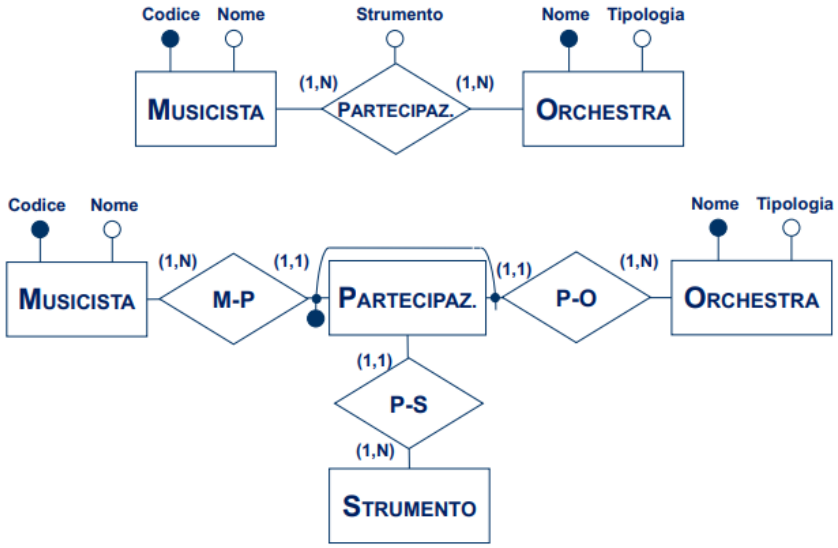
4.4.5 Reificazione di relazione ricorsiva

Nell'esempio, partita potrebbe essere vista come una relazione binaria tra squadra e se stessa.

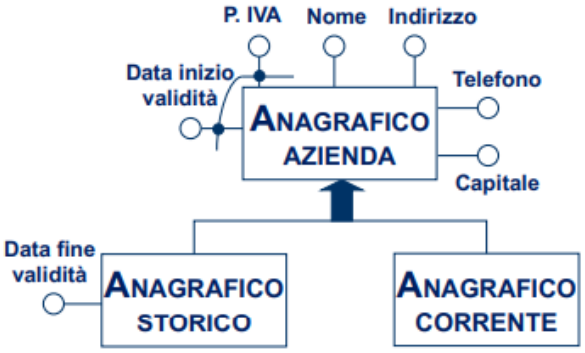
Come sappiamo una squadra può giocare più volte, reificare partita come nello schema seguente è più utile per esprimere questo concetto:

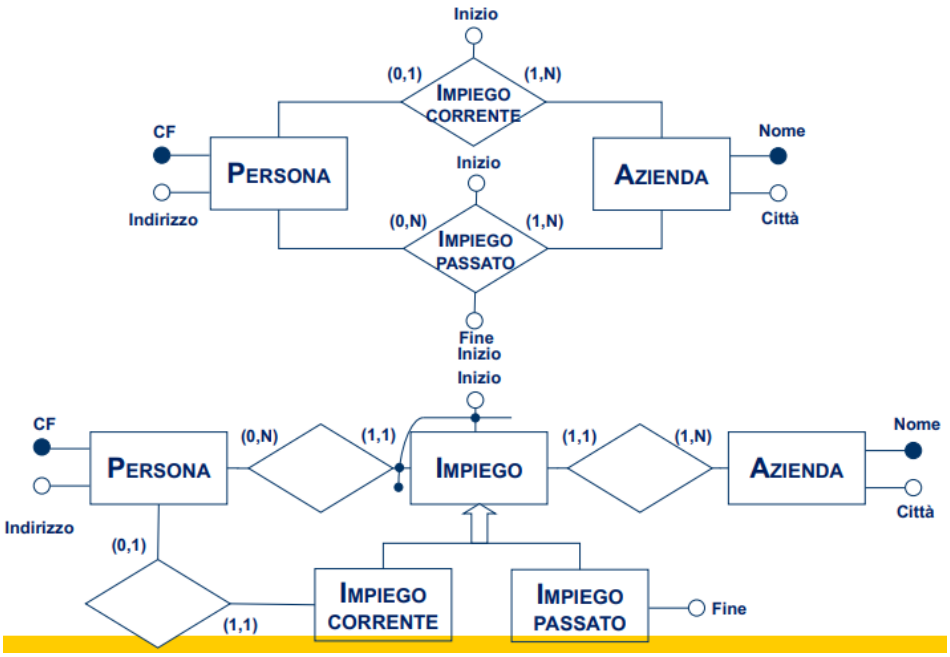


4.4.6 Reificazione di attributo di relazione

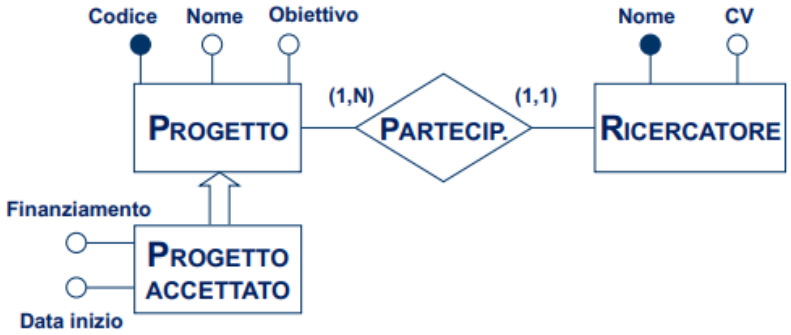


4.4.7 Storicizzazione di concetto

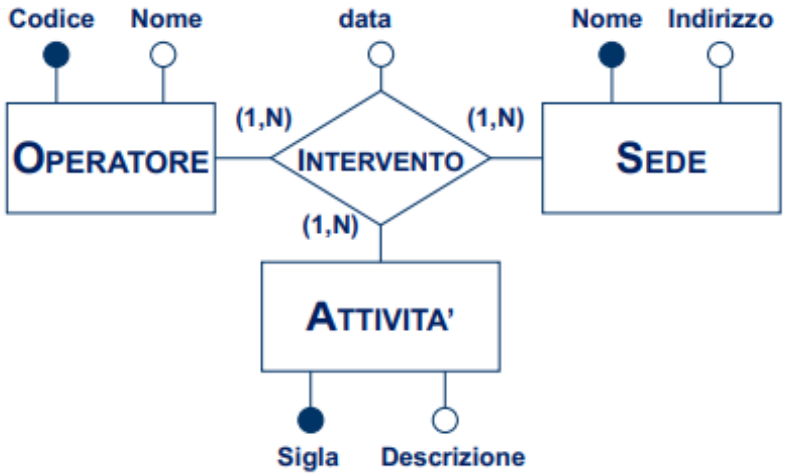


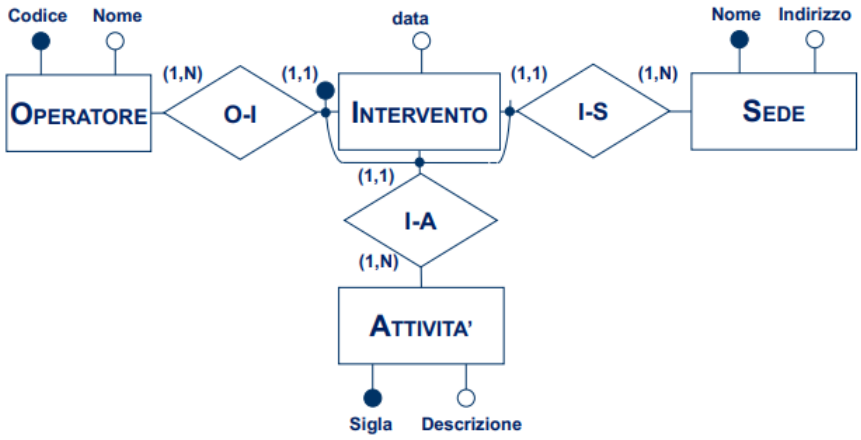


4.4.8 Evoluzione di concetto



4.4.9 Reificazione di relazione ternaria





4.5 Strategie di progetto

Come procediamo con tante specifiche anche dettagliate? Come ci orizzontiamo? Le varie strategie di progetto che possiamo adottare sono:

- **Top-down**: si parte da uno schema iniziale molto astratto ma completo, che viene successivamente raffinato fino ad arrivare allo schema finale
- **Bottom-up**: si suddividono le specifiche in modo da sviluppare semplici schemi parziali ma dettagliati, che poi vengono integrati tra loro
- **Inside-out**: lo schema si sviluppa "a macchia d'olio", partendo dai concetti più importanti, aggiungendo quelli ad essi correlati e così via

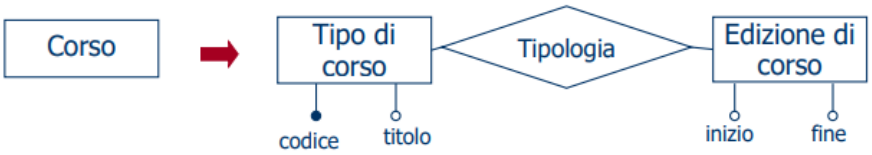
Ciascuna strategia prevede opportune **primitive di raffinamento** che specificano in che modo sostituire o integrare una parte dello schema con una versione più raffinata della stessa. Vediamo i vantaggi e gli svantaggi dei vari approcci:

- **Top-down**
  - **Pro**: non è inizialmente necessario specificare i dettagli
  - **Contro**: richiede sin dall'inizio una visione globale del problema, non sempre ottenibile in casi complessi
- **Bottom-up**:
  - **Pro**: permette una ripartizione delle attività
  - **Contro**: Richiede una fase di integrazione
- **Inside-out**:
  - **Pro**: non richiede passi di integrazione
  - **Contro**: Richiede ad ogni passo di esaminare tutte le specifiche per trovare i concetti non ancora modellati

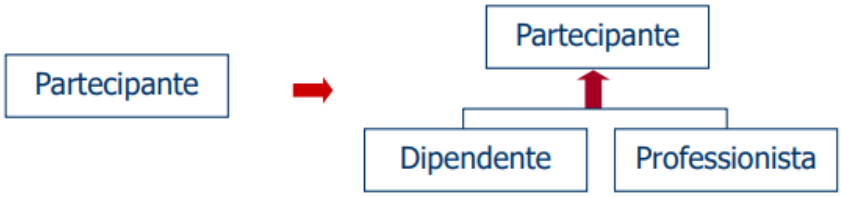
4.5.1 Strategia top-down

Vediamo quali sono le **primitive di trasformazione top-down**: esse operano su un singolo concetto trasformandolo in una struttura più complessa in grado di descrivere il concetto di partenza con maggior dettaglio:

- **T1**: Una entità descirve due concetti diversi legati logicamente fra loro

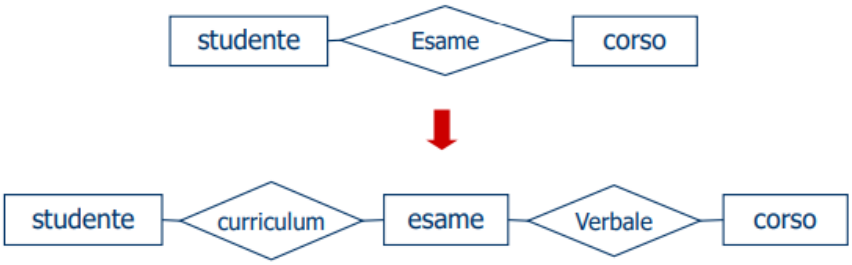


- **T2**: Una entità è composta da sotto-entità distinte

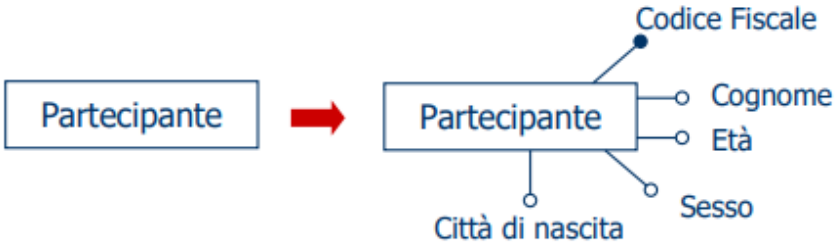


- **T3**: Una relazione in realtà descrive due relazioni diverse fra le stesse entità

- **T4**: Una relazione descrive in realtà un concetto con esistenza autonoma

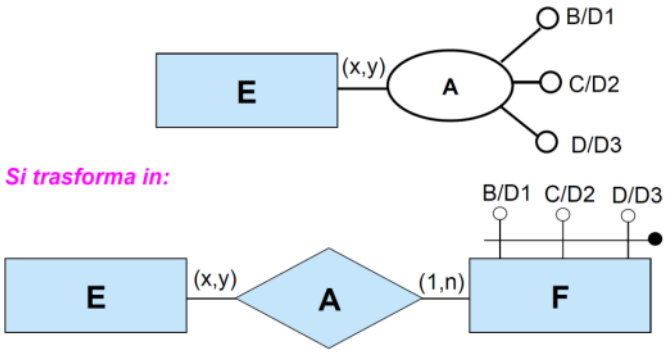


- **T5**: Si aggiungono attributi ad entità

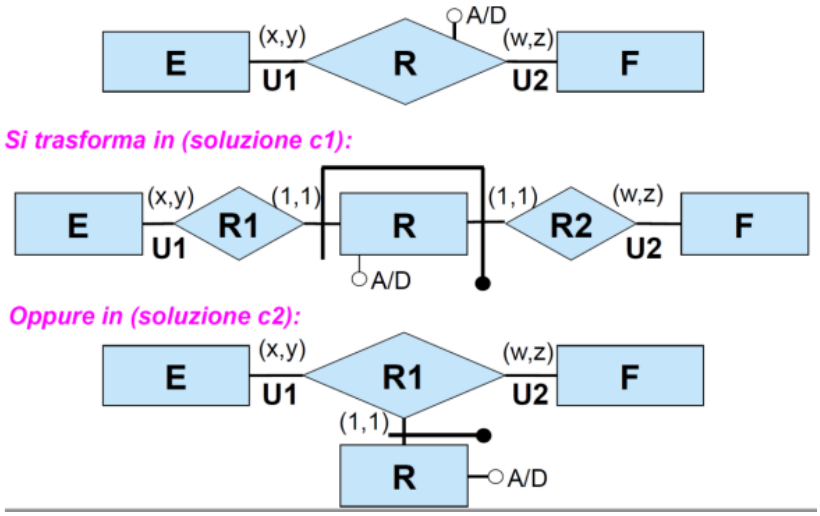


- **T6**: Si aggiungo attributi a relazioni

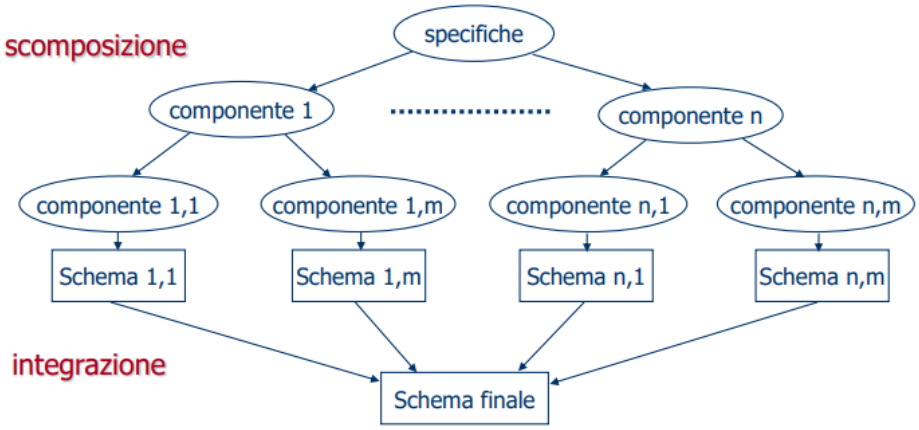
Nella strategia top-down, si trasformano gli attributi composti di un'entità in un'altra entità connessa alla prima da una relazione



Si può anche andare a trasformare una relazione in una entità nel seguente modo:



4.5.2 Strategia bottom-up



Vediamo le **primitive di trasformazione bottom-up**: esse introducono nuovi concetti in grado di descrivere aspetti non ancora rappresentati.

- **T1**: Creazione di una entità relativa a una classe di oggetti con proprietà comuni

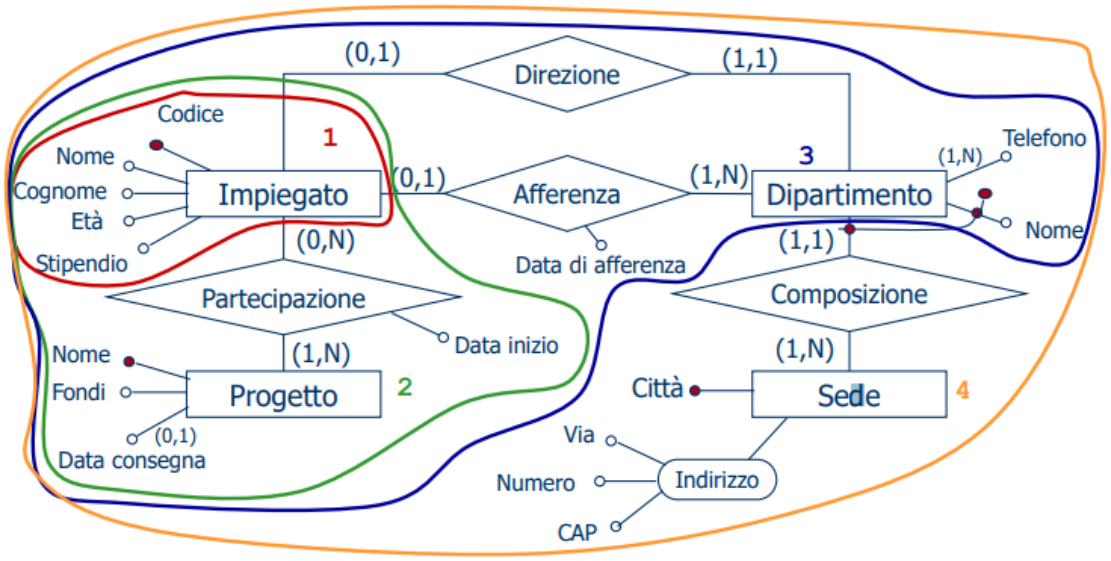
- **T2**: Individuazione di un legame logico fra due entità (relazione)

- **T3**: Individuazione di un legame tra diverse entità riconducibile ad una generalizzazione

- **T4**: A partire da una serie di attributi si individua un'entità che aggrega tali attributi

- **T5**: A partire da una serie di attributi si individua una relazione che aggrega tali attributi

4.5.3 Strategia inside-out



4.5.4 In pratica - Strategia ibrida

Nella pratica si utilizzano diverse tecniche provenienti da strategie diverse. La strategia ibrida combina i vantaggi delle strategie top-down e bottom-up:

- Suddivisone dei requisiti in **componenti separate**
- Definizione di uno **schema scheletro** per i concetti principali

Lo schema scheletro facilita le fasi di integrazione:

- Si individuano i concetti più importanti (**bottom-up**)
- Si organizzano tali concetti in un semplice schema concettuale (**inside-out**)
- Ci si concentra sugli aspetti essenziali (molti attributi, le cardinalità delle relazioni, le gerarchie articolate sono rimandate) (**top-down**)

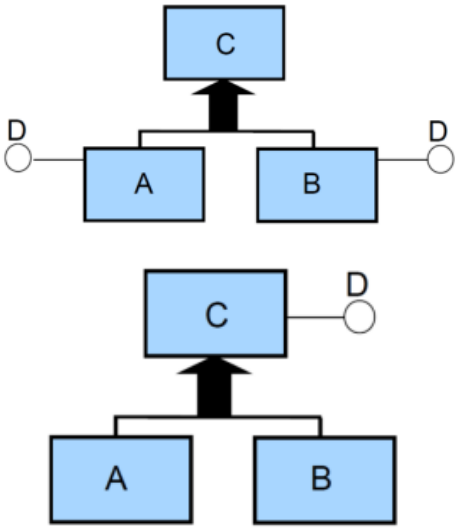
4.6 Qualità di uno schema concettuale

Le qualità di uno schema concettuale sono:

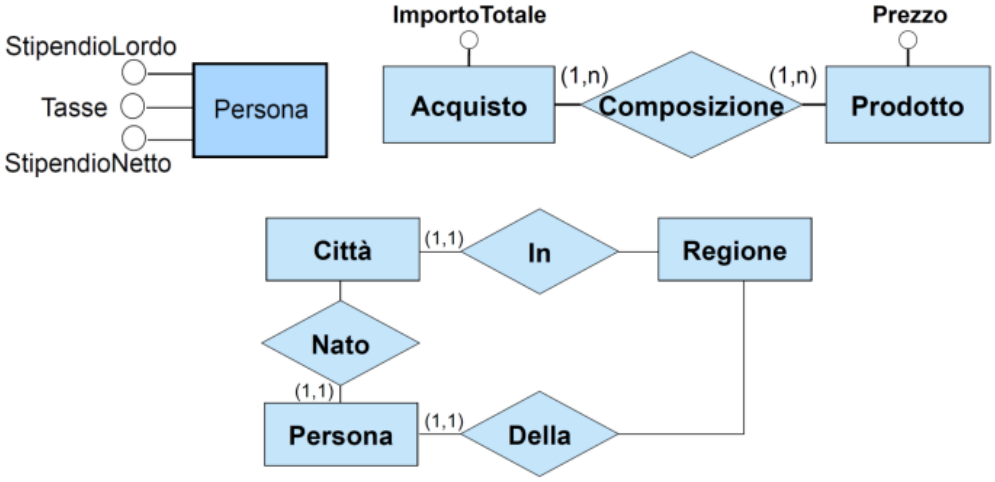
- **Correttezza**: Uso corretto dei costrutti (sintassi e semantica)
- **Completezza**: Tutti i dati sono rappresentati; tutte le operazioni possono essere eseguite (tutti i dati previsti da un'operazione sono raggiungibili *navigando* il diagramma E-R)
- **Leggibilità**: Lo schema deve essere il più possibile autoesplicativo (nomi, layout dello schema)
- **Minimalità**: Lo schema non contiene ridondanze a livello intensionale ed estensionale

4.6.1 Minimalità

Vi sono due tipi di ridondanza: **intensionale** ed **estensionale**. Esempio di ridondanza estensionale e della sua risoluzione è la seguente:



Esempio di ridondanza a livello estensionale è il seguente:



Lo schema non deve contenere ridondanza (concetti che possono essere derivati da altri)