Basi di Dati

UniShare

Davide Cozzi
@dlcgold

Gabriele De Rosa
@derogab

Federica Di Lauro
@f_dila

Indice

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

E-R

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. Una **base di dati** *è un insieme organizzato di dati utilizzato per il supporto allo svolgimento di attività.* L'**informatica** è la *scienza del trattamento relazionale, specialmente per mezzo di macchine automatiche, dell'informazione, considerata come supporto alla conoscenza umana e all'informazione.* Un **sistema informativo** è *un componente di un'organizzazione che gestisce le informazioni d'interesse.* Si divide in:

- acquisizione e memorizzazione
- aggiornamento
- interrogazione
- elaborazione

Un sistema informativo è una porzione automatizzata del sistema informatico. Un sistema informatico gestisce il sistema informativo in modo automatizzato, ne garantisce la memorizzazione, l'aggiornamento dei dati per riflettere le loro variazioni, l'accessibilità dei dati. Un sistema informatico può essere distribuito sul territorio. Le informazioni vengono gestite in vari modi, mediante idee formali, con il linguaggio naturale, graficamente con schemi e con numeri o codici. Anche il mezzo può variare dalla carta ai dispositivi elettronici. Pian piano si è arrivato ad avere una codifica standard per l'informazione, a fini organizzativi. Le informazioni nei sistemi informatici sono rappresentate in modo essenziale mediante i **dati**, questa è la *sintassi*. La *semantica* invece si ottiene con le intestazioni delle tabelle. I dati sono una risorsa strategica perché sono stabili nel tempo. Solitamente i dati sono immutati durante una migrazione tra un sistema e un altro. Un **Data Base** è una collezione di dati usati per rappresentare un'informazione di interesse di un sistema informativo. Un **DBMS** è un software che gestisce un database. Un base di dati è anche un insieme di archivi in cui ogni dato è rappresentato logicamente una e una sola volta è può essere usato da un insieme di applicazioni e da più utenti su vari criteri di riservatezza. Si hanno le seguenti caratteristiche:

- 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 minima è controllata ed è assicurata la consistenza delle informazioni
- i dati sono disponibili per utenze diverse e concorrenti
- i dati sono controllati e protetti da malfunzionamenti hardware e software
- i dati sono indipendenti dal programma

Si hanno tre fasi di creazione di una tabella:

1. definizione
2. creazione e popolazione

3. manipolazione

Si garantiscono:

- privatezza
- affidabilità
- efficienza
- efficacia

Un organizzazione è divisa in vari settori e ogni settore ha un suo sottosistema informativo (non necessariamente disgiunto). Le basi dati solitamente sono condivise. Una base di dati gestisce i meccanismi di autorizzazione e ha un controllo della concorrenza, oltre ad essere una risorsa integrata e condivisa. Una base di dati deve essere conservata a lungo termine e si ha una gestione delle **transazioni**. Una **transazione** *è un insieme di operazioni da considerare indivisibile, "atomico", corretto anche in presenza di concorrenza e con effetti definitivi.* La sequenza di operazioni sulla base di dati deve essere eseguita nella sua interezza (sono quindi atomiche) e l'effetto di transazioni concorrenti deve essere coerente. La conclusione positiva di una transazione corrisponde ad un impegno, *commit*, a mantenere traccia del risultato in modo definitivo, anche in presenza di guasti e di esecuzione concorrente.
I DBMS devono essere efficienti, utilizzando al meglio memoria e tempo. Devono anche essere efficaci e produttivi.
Si hanno delle caratteristiche nell'apporoccio alla base dati:

- natura autodescrittiva di un sistema di basi di dati: il sistema di basi di dati memorizza i dati e anche una descrizione completa della sua struttura (catalogo) queste informazioni sono chiamate metadati. Questo consente al DBMS di lavorare con qualsiasi applicazione
- separazione tra programmi e dati: chiamata indipendenza tra programmi e dati. E' 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 vista concettuale del database
- supporto di viste multiple dei dati: 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. Permette a più utenti concorrenti di accedere contemporaneamente al database. Il controllo della concorrenza garantisce che più utenti impegnati ad aggiornare gli stessi dati lo facciano in maniera controllata: il risultato degli aggiornamenti è corretto

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, il significato (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 "concreto", che varia nel tempo al variare della situazione di ciò che stiamo descrivendo)

Si hanno due tipi di modelli:

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

ecco l'architettura di un DBMS:

$$\begin{array}{c} utente \\ \downarrow \\ schema\ \ logico \\ \downarrow \\ schema\ \ interno \\ \downarrow \\ database \end{array}$$

Si hanno 2 schemi:

1. **schema logico:** descrizione della base di dati nel modello logico (ad esempio, la struttura della tabella)

2. **schema interno (o fisico):** rappresentazione dello schema logico per mezzo di strutture memorizzazione (file; ad esempio, record con puntatori, ordinati in un certo modo)

Il livello logico è indipendente da quello fisico infatti una tabella è utilizzata nello stesso modo qualunque sia la sua realizzazione fisica (che può anche cambiare nel tempo). **Noi tratteremo solo il livello logico**. Ecco i tre schemi dell'architettura ANSI/SPARC:

1. **schema logico:** descrizione dell'intera base di dati nel modello logico "principale" del DBMS

2. **schema fisico:** rappresentazione dello schema logico per mezzo di strutture fisiche di memorizzazione

3. **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 (che a volte coincide con quello logico) e si hanno 2 forme di indipendenza:

1. **indipendenza fisica**

2. **indipendenza logica**

Il livello logico e quello esterno sono indipendenti da quello fisico.
Un linguaggio interattivo per gestire una base dati è SQL, che funziona tramite **query**. Si hanno linguaggi per la definizione di dati, i DLL, *Data Definition Languages*, e linguaggi di manipolazione dei dati, DML, *Data Manipulation Languages*. I primi definiscono e i secondi interrogano e aggiornano. SQL può svolgere le operazioni di entrambi.
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:

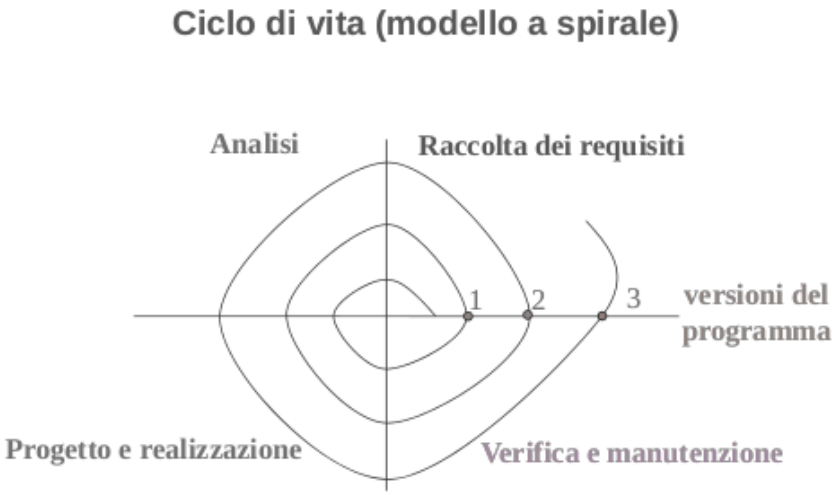
- progettisti e realizzatori di DBMS

- progettisti della base di dati e amministratori della base di dati (DBA), persona o gruppo di persone responsabile del controllo centralizzato e della gestione del sistema, delle prestazioni, dell'affidabilità, delle autorizzazioni. Le funzioni del DBA includono quelle di progettazione, anche se in progetti complessi ci possono essere distinzioni
- progettisti e programmatori di applicazioni

Si hanno le seguenti fasi del ciclo di vita:

- **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

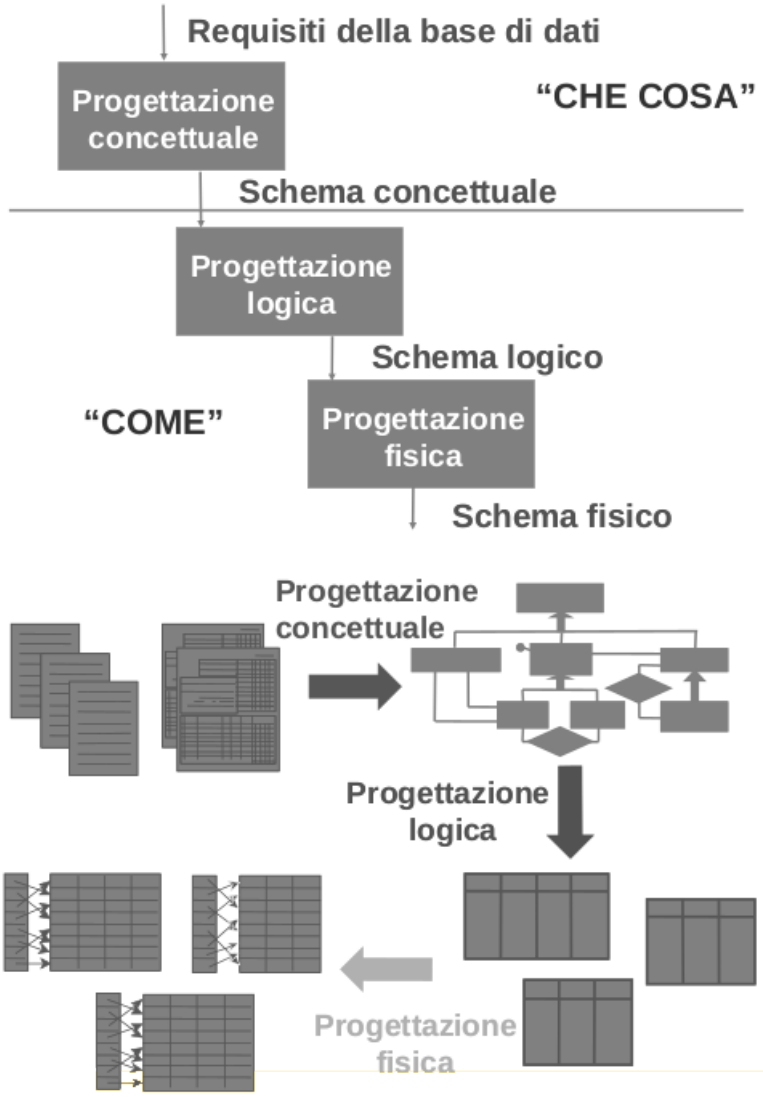
ogni fase può tornare a quella precedente in caso di problemi. Prima di implementare una base di dati bisogna avere delle specifiche ben definite.
Si ha quindi un ciclo di vita con **modello a spirale**, in quanto ogni software deve avere delle releases, e per ognuna si ripete il ciclo di vita.
In questo processo a rimanere stabili sono prettamente i **dati**. Prima si progetta la base dati, con una **metodologia di progetto**, e poi l'applicazione:



1 Una **metodologia** è un'articolazione in fasi di guida ad un'attività di progettazione. Per una base dati serve una metodologia che:

- 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

Si separa il cosa rappresentare e il come farlo.
Si parte dalla progettazione concettuale, si passa a quella logica e si finisce con quella fisica.



Ognuna di queste 3 fasi si basa su un modello che permette di generare una rappresentazione formale, uno schema, delle basi di dati ad un dato livello di astrazione: schema concettuale, schema logico e schema fisico.

- 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 con 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 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 dell'hw/sw scelto. Il risultato è lo schema fisico che descrive le strutture di memorizzazione ed accesso ai dati

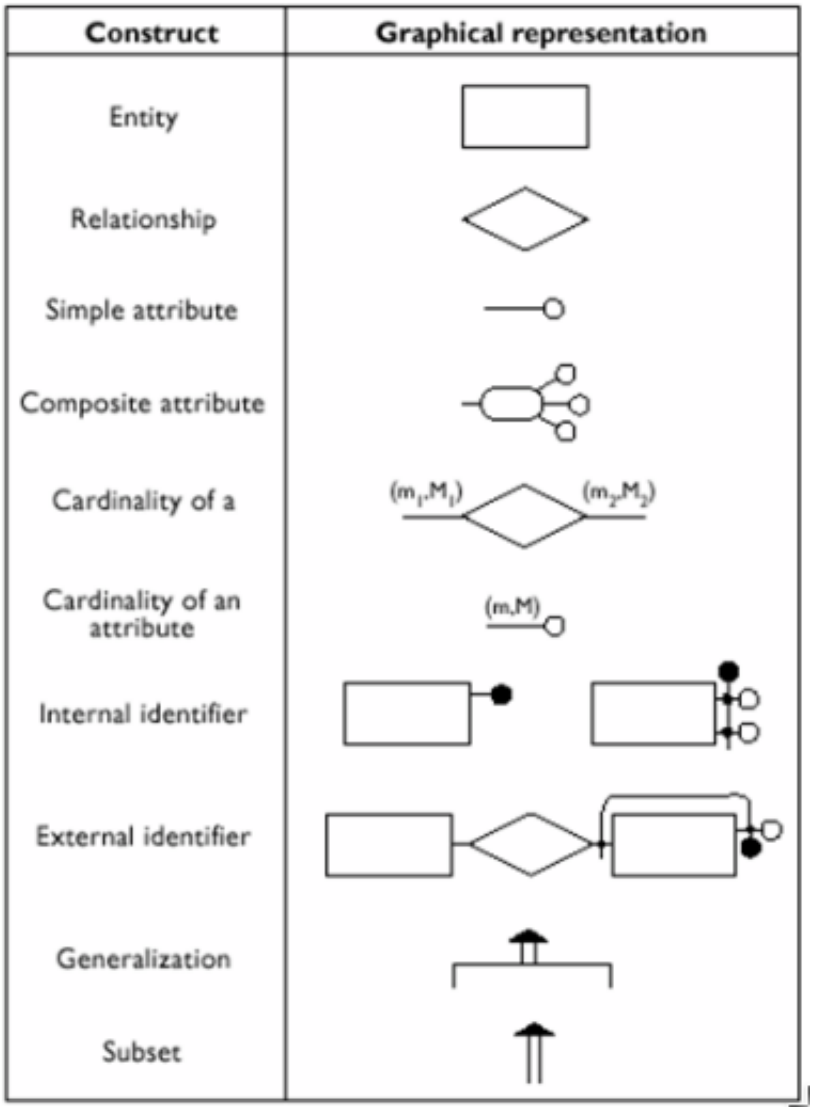
si ha quindi:

Nel nostro caso si ha che il modello E-R è lo schema concettuale e lo schema logico è dato dal modello razionale.
Si hanno dei vantaggi con la progettazione concettuale, prevale infatti l'aspetto intensionale indipendente dalla tecnologia ed è una rappresentazione grafica. È utile per la documentazione.
In uno schema E-R abbiamo le entità, in costrutti, rappresentati da rettangoli e le relazioni da rombi, con indicata l'arità (la cardinalità) (come in un UML). Esistono varie versioni del modello Entità-Relazione *ER.* si hanno i seguenti costrutti:

- entità
- relazione
- attributo semplice
- attributo composto
- cardinalità
- cardinalità di un attributo

- identificatore interno
- identificatore esterno
- generalizzazione
- sottoinsieme

così rappresentati:



Un entità è una classe di oggetti dell'applicazione di interesse con proprietà comune con esistenza "autonoma" e della quale si vogliono specificare fatti specifici. Ogni entità ha un nome che la identifica univocamente, con le seguenti specifiche:

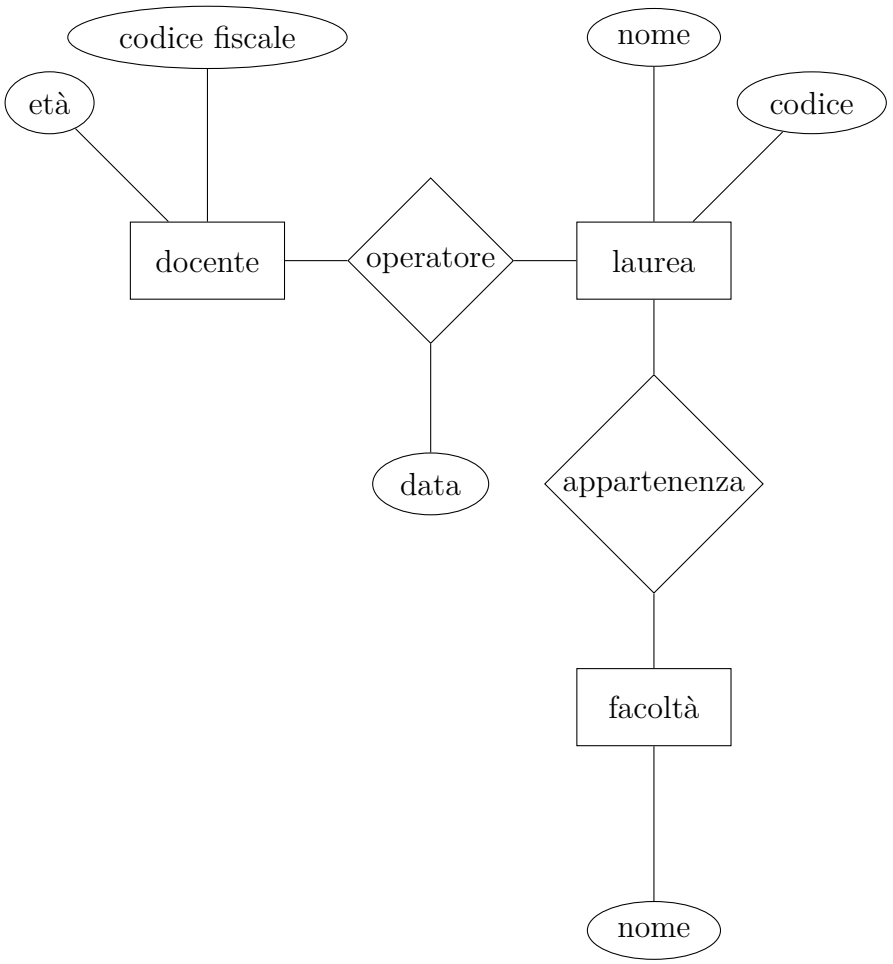
- si hanno nomi espressivi
- si usa il singolare

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. Ovviamente un elemento della classe ha degli attributi, rappresentati da dei pallini collegati alla classe. Un attributo ha un dominio che però non viene rappresentato nell'E-R ma nella documentazione. Entità e attributi funzionano con determinate regole, un attributo è una funzione:

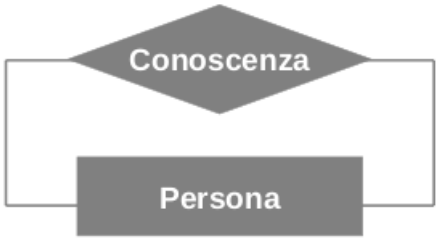
- un enità non può avere valori diversi della stesso attributo
- due entità possono avere lo stesso valore di un attributo
- bisogna sempre specificare il valore di un attributo, si ha una funzione totale

Si ha che gli **attributi composti** si ottengono raggruppando attributi di una medesima entità o relazione che presentano 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'associazione, o relazione, è un fatto che descrive un'azione o una situazione e che stabilisce legami logici tra istanze di entità (associa, mette in relazione) nella realtà che stiamo considerando. Si possono avere legami tra più entità e il numero di entità coinvolte in una relazione ne determina il **grado**. Valgono le stesse regole delle entità per dare un nome alle relazioni (nomi espressivi e al singolare) ma se ne aggiunge una: si usano sostantivi e non verbi.
Un'**istanza di associazione** è un a combinazione o aggregazione di istanze di entità che prendono parte all'associazione (per esempio "prof. Schettini" è istanza di associazione per l'entità docente). Dalla semantica delle relazioni segue immediatamente che non possono esistere due istanze della stessa relazione che coinvolgono le stesse istanze di entità.
Due entità possono essere coinvolte in più relazioni
Le relazioni possono avere attributi, con valori specificati in un certo dominio. Un attributo di una relazione tra più entità modella una proprietà del legame tra tutte le entità rappresentato dalla relazione. Gli attributi delle relazioni si rappresentano come quelli di entità.

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



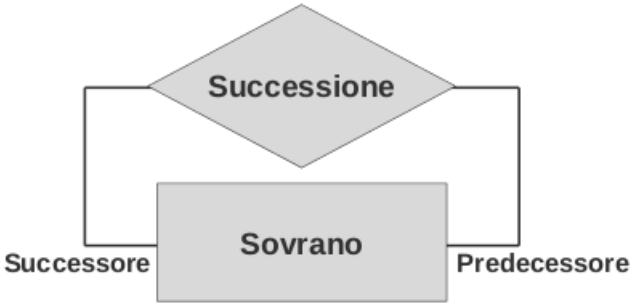
Una associazione può coinvolgere "due o più volte" la stessa entità. Si ha un'*associazione ricorsiva o ad anello*:



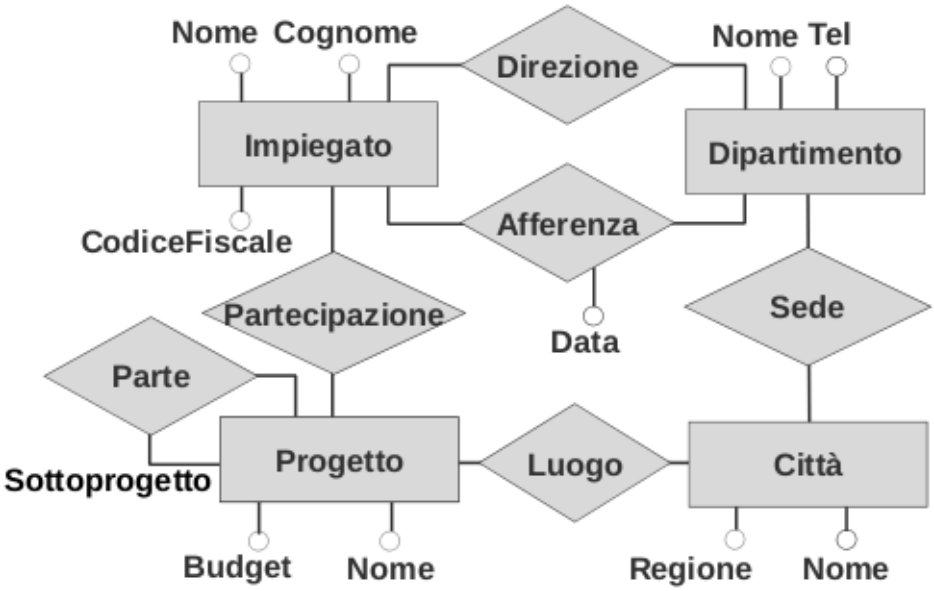
Un'associazione ad anello può essere o meno:

- **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:

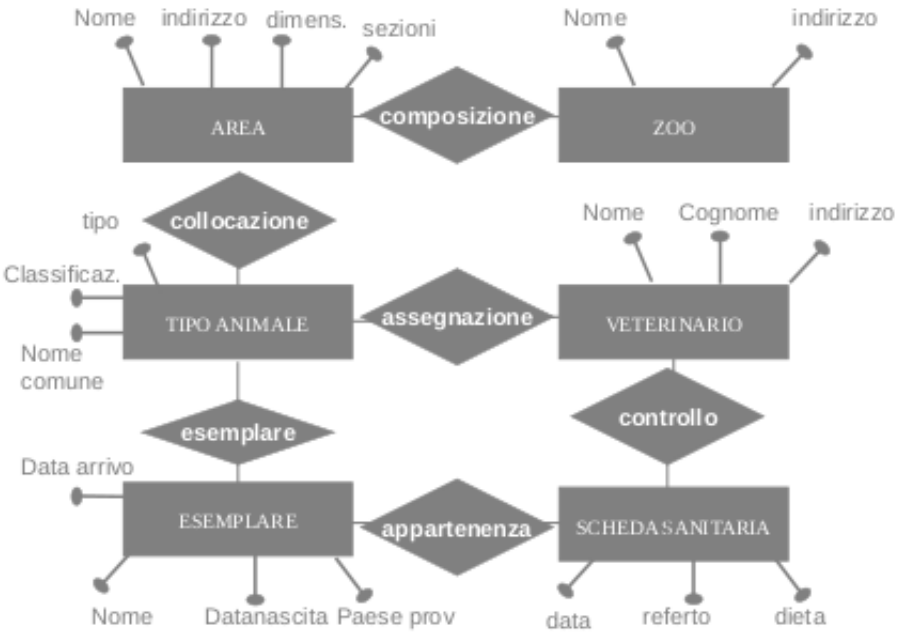
- un'**entità**:
  - se le sue istanze sono concettualmente significative indipendentemente da altre istanze
  - se ha o potrà avere delle proprietà indipendenti dagli altri concetti
  - se ha o potrà avere relazioni con altri concetti
- una **relazione**:
  - se le sue istanze non sono concettualmente 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:

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

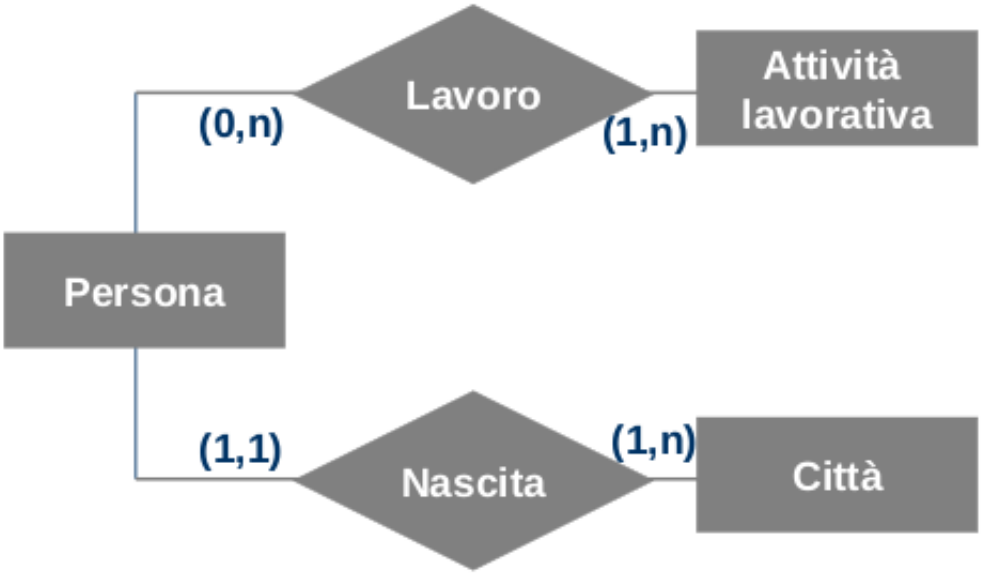
Esempio 2.

- *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

Le relazioni possono avere una **cardinalità**, ovvero una coppia di valori che si associa a ogni entità che partecipa a una relazione che specificano il numero minimo e massimo di occorrenze delle relazione cui ciascuna occorrenza di una entità può partecipare. Si rappresenta con un range tra un min e un max: (min, max)



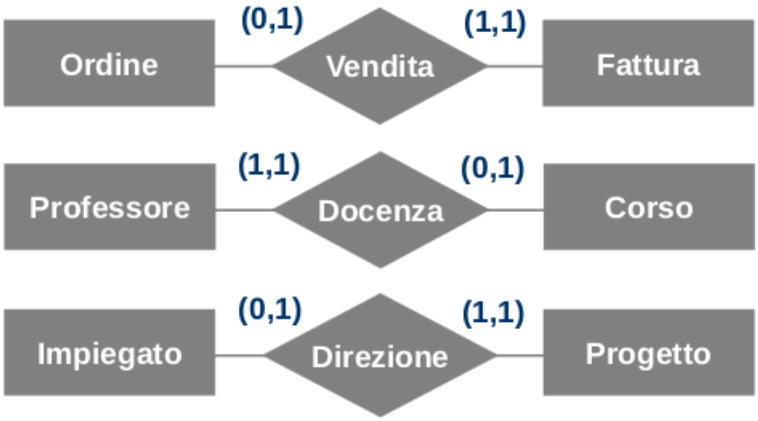
Si hanno tre simboli:

1. 0 per la cardinalità minima indicante una *partecipazione opzionale*
2. 1 per la cardinalità minima indicante una *partecipazione obbligatoria* o indicante una cardinalità massima limitata
3. N indicante una cardinalità massima senza limite

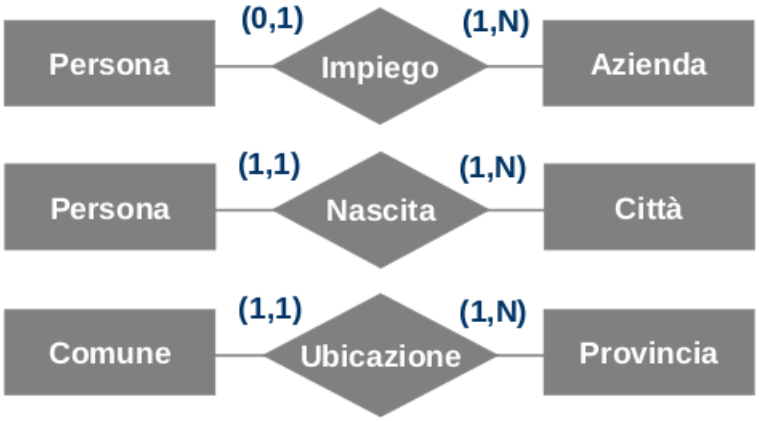
si hanno quindi relazioni:

- **uno a uno** se le cardinalità massime di entrambe le entità sono uno, $(1, 1)$ relazione $(1, 1)$
- **uno a molti**, $(0/1, 1)$ relazione $(1, N)$
- **molti a molti**, $(0/1, N)$ relazione $(0/1, N)$

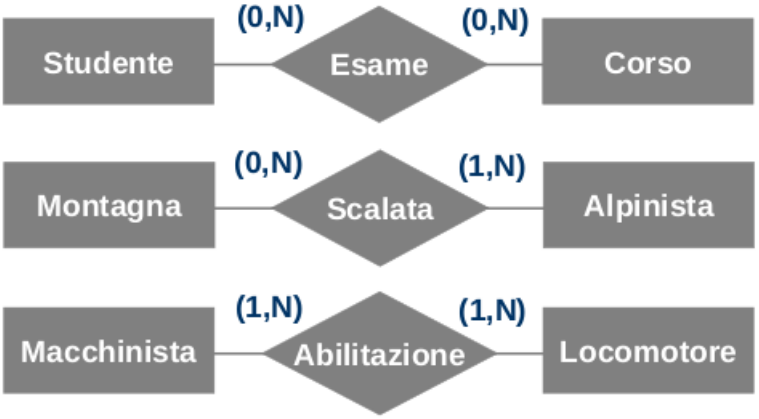
Relazioni "uno a uno"



Relazioni "uno a molti"



Relazioni "molti a molti"



vediamo un esempio:

Esempio 3. *si ha:*

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

Un vincolo di cardinalità tra una entita' 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. La cardinalità minima deve essere minore di quella massima. Ovviamente si possono avere altri simboli:

Istanza dello schema:

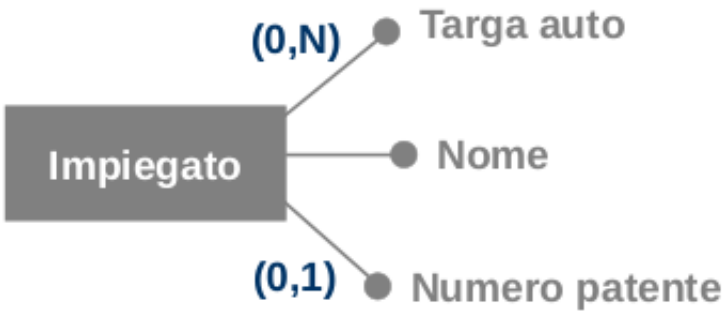
Istanze(Impiegato) = { a,b,c }

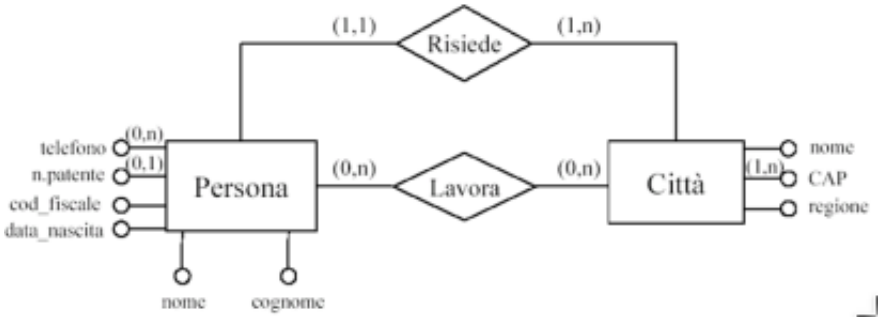
istanze(Progetto) = { x,y,v,w,z }

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

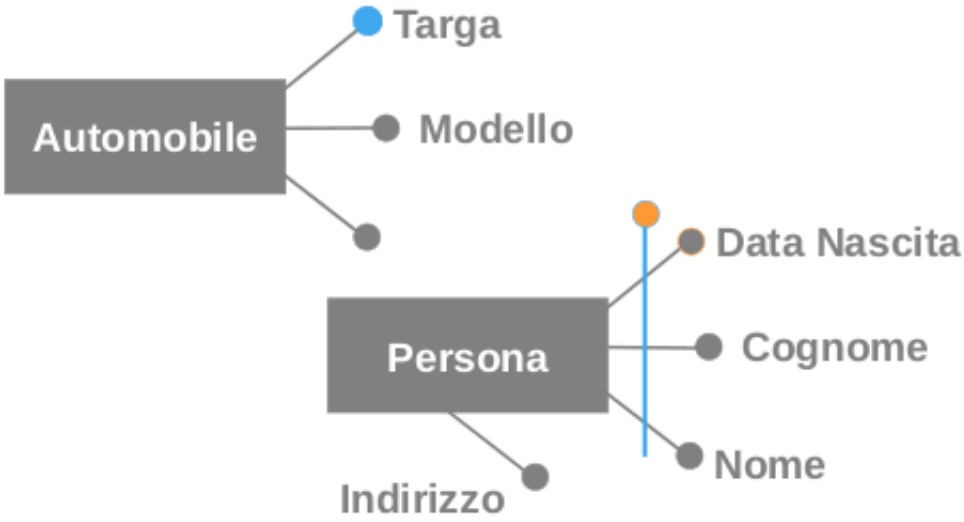
impiegato
Si può mettere cardinalità agli attributi per indicare l'opzionalità o indicare attributi multivalore:



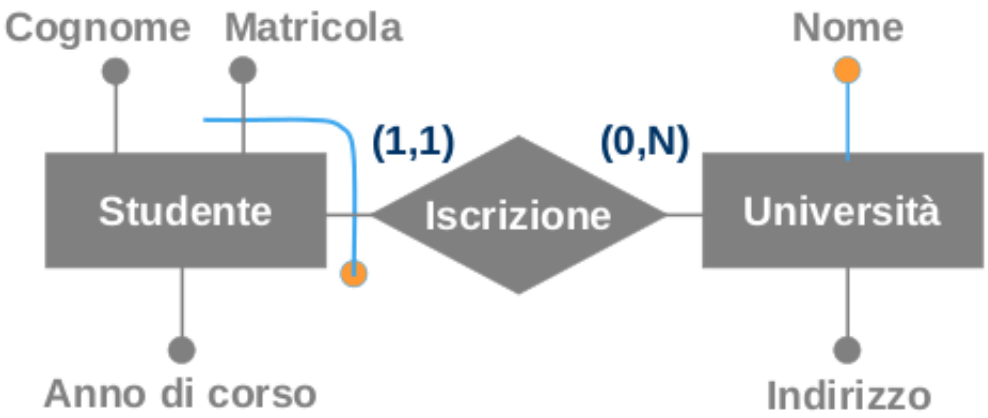


infatti, per esempio, una persona può non avere la patente etc...
Un **identificatore di una entità** permette l'identificazione univoca delle occorrenze di un'entità. È formato da:

- attributi dell'entità, *identificatore interno*, tipo il codice fiscale. Si ha 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



- (attributi +) entità esterne attraverso relationship, *identificatore esterno*, tipo la matricola che dipende dall'università. Come notazione si ha che 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:



Si hanno alcune osservazioni:

- ogni entità deve possedere almeno un identificatore, ma può averne in generale più di uno

- una identificazione esterna è possibile solo attraverso una relationship a cui l'entità da identificare partecipa con cardinalità $(1, 1)$

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