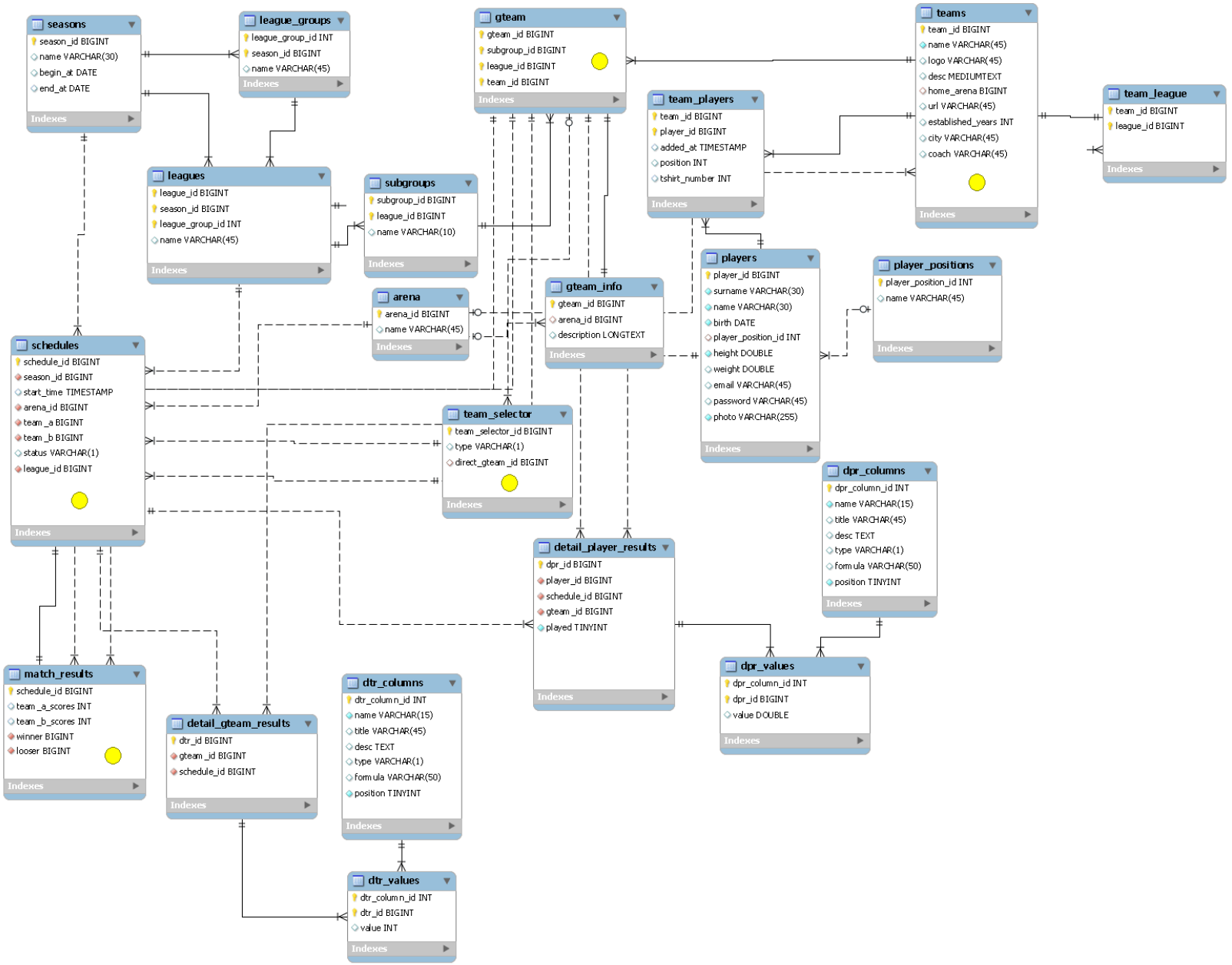
Design di un Database



Design di un database

- Progettare un database implica definire quanto i seguenti aspetti:
  - Struttura
  - Caratteristiche
  - Contenuti
- Il ciclo di design di un database si suddivide in tre fasi principali:
  - progettazione concettuale
  - progettazione logica
  - progettazione fisica
- A sua volta la progettazione di un database è soltanto una delle fasi che costituiscono il ciclo di vita di un sistema informativo:
  - Studio di fattibilità
  - Raccolta ed analisi dei requisiti
  - Progettazione (design)
  - Implementazione
  - Validazione e testing
  - Funzionamento (operatività)

Ciclo di vita del sistema

- **Studio di fattibilità**: vengono analizzati i costi delle possibili soluzioni e si decide la priorità i realizzazione dei vari componenti del sistema.
- **Raccolta ed analisi dei requisiti**: vengono definite le proprietà e le funzionalità che il sistema deve offrire attraverso un'interazione con gli utenti futuri. Il risultato è una descrizione informale dei dati coinvolti e delle operazioni su di essi. Vengono inoltre stabiliti i requisiti dell'hardware e del software.
- **Progettazione**: vi sono due fasi che possono procedere parallelamente o in successione. La prima, i.e., il design del database stabilisce la struttura e l'organizzazione dei dati. La seconda, i.e., il design operazionale stabilisce le caratteristiche delle applicazioni.
- **Implementazione**: in base a quanto stabilito nelle fasi precedenti, viene creato e popolato il database e vengono scritti i programmi applicativi.
- **Validazione e test**: viene effettuata una verifica del corretto funzionamento del sistema.
- **Funzionamento**: il sistema diventa operativo a tutti gli effetti. Questa fase normalmente richiede soltanto interventi di manutenzione.

Ciclo di vita del sistema

- Le fasi descritte precedentemente vengono raramente svolte in sequenza. Infatti capita spesso che durante una fase sorga un problema che porti a dover riconsiderare quanto stabilito precedentemente.
- Inoltre è spesso presente un'ulteriore fase chiamata *prototyping*, che consiste nel realizzare rapidamente delle versioni semplificate del sistema (prototipi) per verificarne alcune funzionalità.
- Nonostante il database sia soltanto una delle componenti di un sistema informativo, l'importanza fondamentale che hanno i dati spinge a riservare un'attenzione particolare alla progettazione della base di dati, lasciando in secondo piano le applicazioni che poi la utilizzeranno.

Metodologia di design

- Una buona metodologia di progettazione di un database deve prevedere quanto segue:
  - una decomposizione dell'attività complessiva di design in una successione di fasi indipendenti l'una dall'altra;
  - delle strategie da seguire nelle varie fasi di progettazione e dei criteri che consentano di effettuare una scelta nel caso in cui vi siano più alternative valide da seguire;
  - dei modelli di riferimento per descrivere input ed output (i.e., prerequisiti e risultati) delle varie fasi.
- Le proprietà che una buona metodologia deve garantire sono le seguenti:
  - generalità rispetto al sistema ed alle applicazioni in uso,
  - qualità del prodotto (accuratezza, completezza ed efficienza),
  - facilità di utilizzo delle strategie e dei modelli di riferimento.

Metodologia di design

- Nel campo di ricerca sui database si è consolidata una metodologia che soddisfa tutte le proprietà precedentemente descritte.
- Il principio su cui si basa è effettuare una chiara divisione fra le decisioni che riguardano *cosa* rappresentare e quelle che riguardano *come* farlo.
- Le fasi consecutive in cui è suddivisa la metodologia sono le seguenti:
  - Progettazione concettuale: il modello di dati utilizzato è quello *concettuale*, che permette di descrivere l'organizzazione dei dati ad un alto livello di astrazione, producendo come risultato uno schema concettuale.
  - Progettazione logica: lo schema concettuale viene tradotto in uno schema logico, adottando un *modello di dati logico* che sia supportato dal DBMS scelto.
  - Progettazione fisica: lo schema logico viene completato con tutti i dettagli necessari per l'implementazione fisica sul DBMS scelto. Il risultato è uno schema fisico dipendente da un *modello fisico* strettamente dipendente dalle caratteristiche specifiche (e.g. strutture per l'organizzazione dei dati) del DBMS scelto.

Requisiti

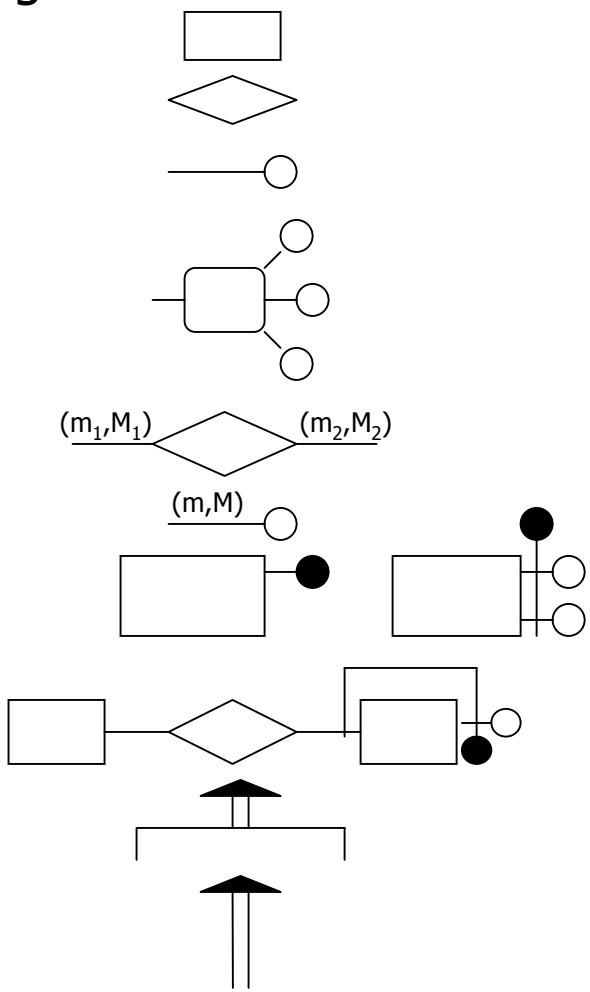
- I requisiti possono essere suddivisi in
  - Requisiti dei dati (riguardano il contenuto del database)
  - Requisiti operazionali (riguardano l'uso del database da parte di utenti e programmi)
- Per quanto riguarda la fase di progettazione concettuale ciò che conta sono i requisiti dei dati, mentre i requisiti operazionali sono utili soltanto per verificare che lo schema concettuale prodotto sia completo.
- Nella fase di progettazione logica lo schema concettuale ricevuto in input rappresenta i requisiti dei dati, mentre i requisiti operazionali sono utilizzati per costruire lo schema logico in modo da garantire un'esecuzione efficiente delle operazioni stesse.
- Nella fase di progettazione fisica lo schema logico ricevuto in input ed i requisiti operazionali sono utilizzati per ottimizzare l'efficienza del sistema informativo in base alle caratteristiche del particolare DBMS scelto.

Il modello Entità-Relazione

- Il modello Entità-Relazione (E-R) è un modello di dati concettuale.
- Fornisce dei costrutti per descrivere i requisiti dei dati di un'applicazione in modo indipendente da come vengano organizzati e gestiti i dati in un sistema.
- Ad ogni costrutto è associata una rappresentazione grafica.
- In questo modo è possibile rappresentare tramite dei diagrammi (ottenuti combinando i costrutti di base) i requisiti che i dati devono soddisfare.

Costrutti del modello E-R

- I costrutti del modello E-R sono i seguenti:
  - Entità
  - Relazione
  - Attributo semplice
  - Attributo composto
  - Cardinalità di una relazione
  - Cardinalità di un attributo
  - Identificatore interno
  - Identificatore esterno
  - Generalizzazione
  - Sottoinsieme

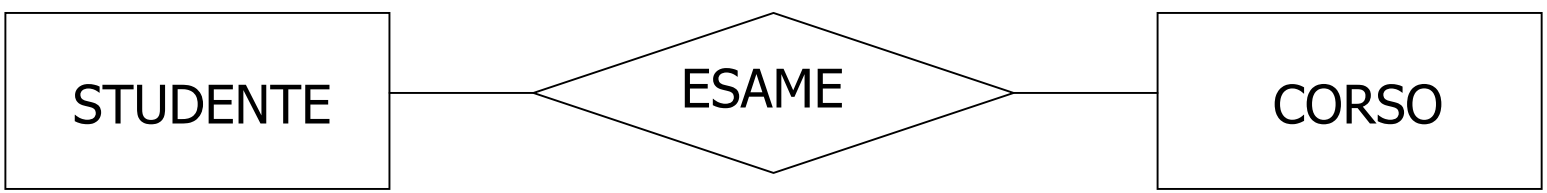


Entità

- Le entità rappresentano insiemi di oggetti (studenti, impiegati, città ecc.) che hanno delle proprietà comuni ed esistono autonomamente.
- Un'occorrenza di un'entità è un oggetto dell'insieme che quest'ultima rappresenta (e.g., la città di Roma è un'occorrenza dell'entità Città).
- Un'occorrenza di un'entità non è un valore che identifichi l'oggetto, ma l'oggetto stesso. Quindi un'occorrenza di un'entità esiste indipendentemente dalle sue proprietà.
- In uno schema ogni entità ha un nome univoco ed è rappresentata da un rettangolo contenente quest'ultimo:

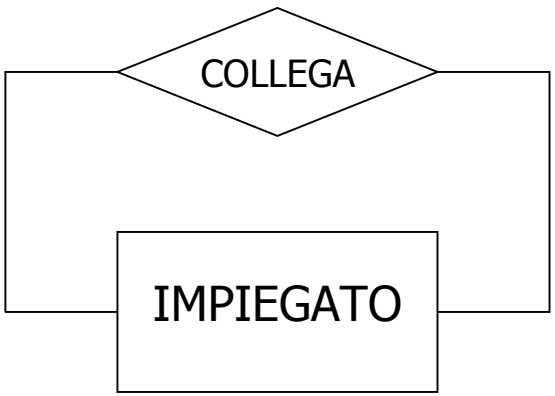
Relazioni

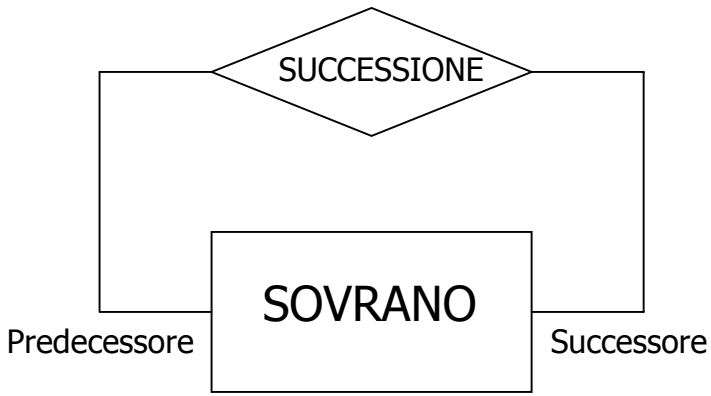
- Le relazioni rappresentano corrispondenze logiche fra due o più entità.
- Un'occorrenza di una relazione che coinvolge n entità è una n-tupla in cui ogni elemento è un'occorrenza dell'entità corrispondente. Quindi un'occorrenza di una relazione binaria è una coppia di elementi di cui il primo è un'occorrenza della prima entità, mentre il secondo è un'occorrenza della seconda entità.
- Esempi di possibili relazioni sono la relazione RESIDENZA fra le entità CITTÀ e IMPIEGATO e la relazione ESAME fra le entità STUDENTE e CORSO.
- Graficamente una relazione si rappresenta con un rombo che racchiude il suo nome (univoco all'interno dello schema):



Relazioni ricorsive

- Le relazioni ricorsive collegano un'entità a se stessa:

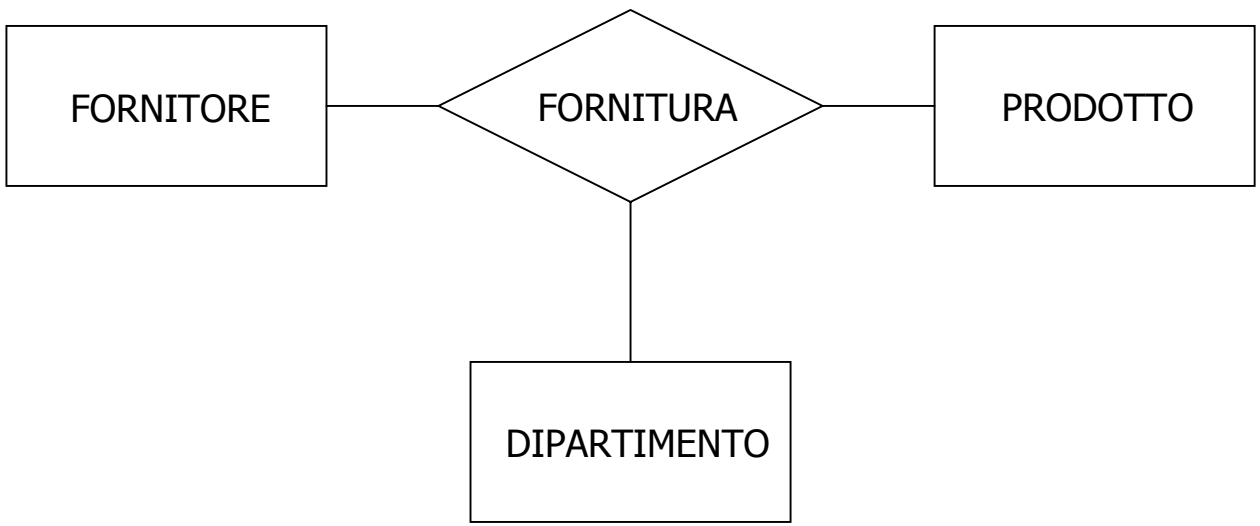




- Nel caso della seconda relazione (SUCCESSIONE), esiste un'asimmetria. Quindi si rende necessario distinguere i due ruoli che l'entità SOVRANO gioca, associando degli identificatori (Predecessore e Successore) alle linee che escono dalla relazione ricorsiva.

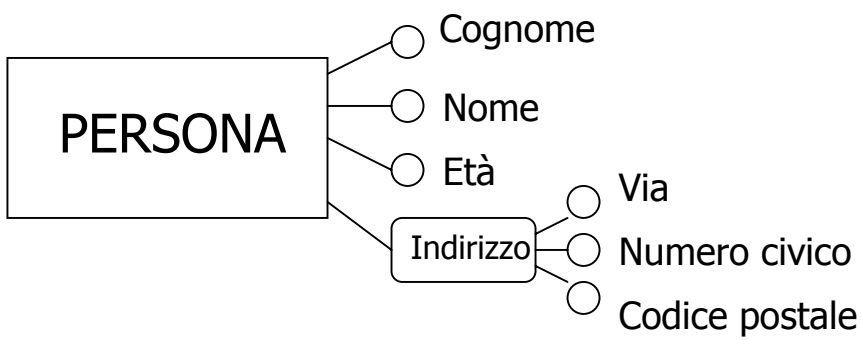
Relazioni fra più di due entità

- Un esempio di relazione fra più di due entità è quella tale che ogni sua occorrenza descrive il fatto che un fornitore fornisce un dato prodotto ad un dipartimento:

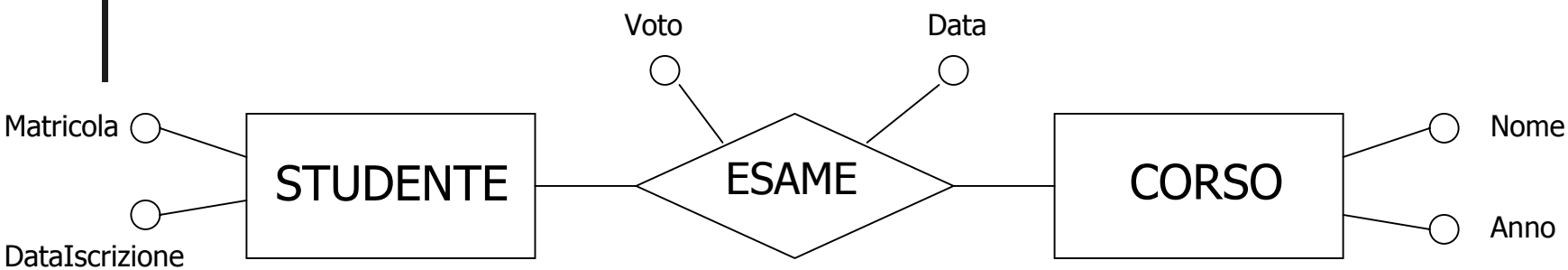


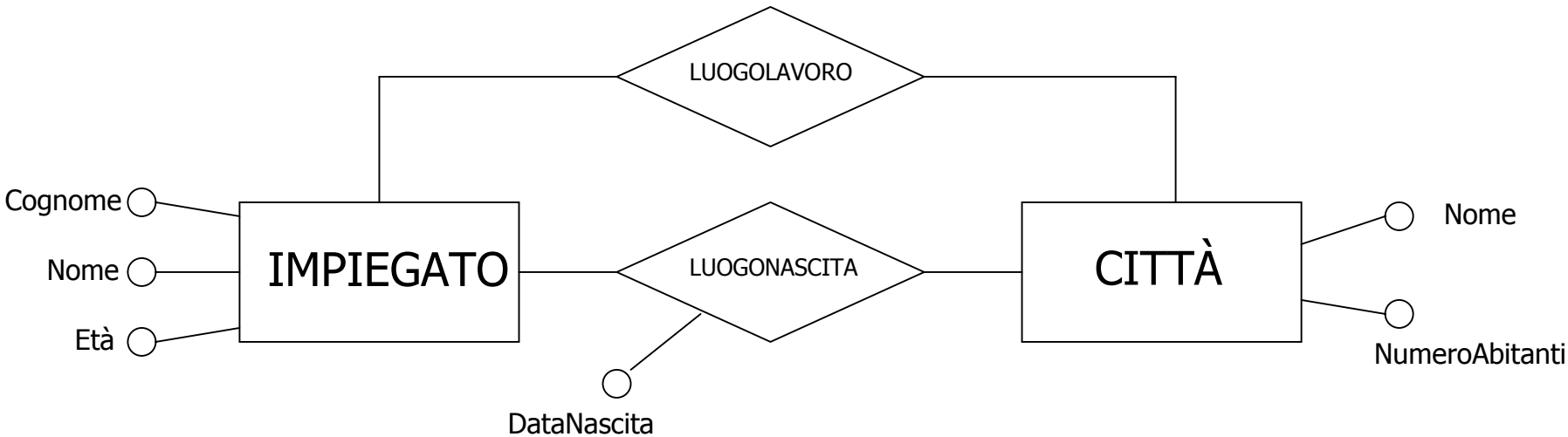
Attributi

- La funzione degli attributi è quella di descrivere le proprietà elementari di entità e relazioni.
- Un attributo associa ad ogni occorrenza dell'entità o relazione su cui è definito un valore appartenente ad un insieme (il dominio dell'attributo).
- A volte può essere conveniente raggruppare diversi attributi della stessa entità o relazione, se sono strettamente correlati. In questo caso si ottiene un attributo composto.

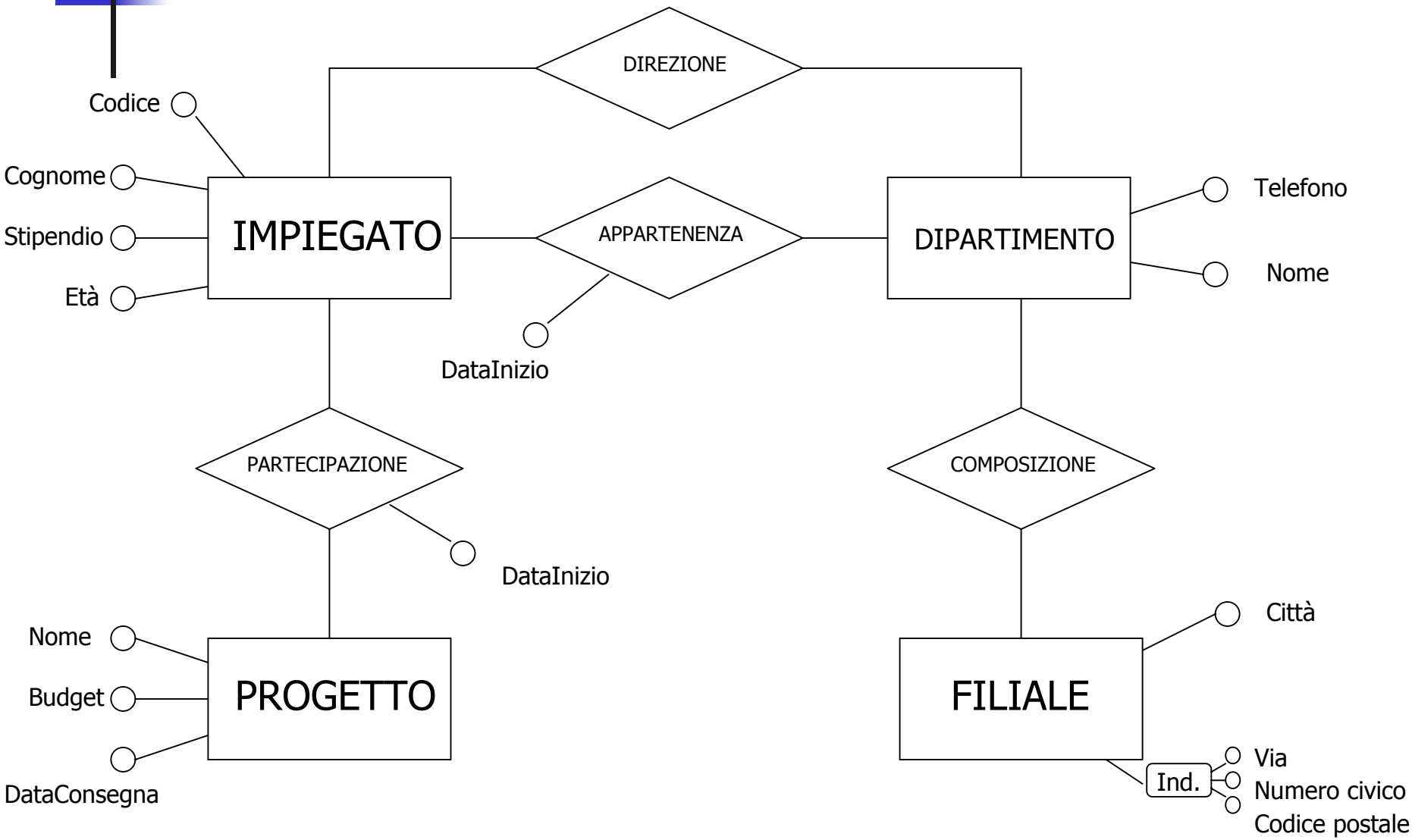


Esempi


Voto
Data
Matricola
STUDENTE
ESAME
CORSO
Nome
DataIscrizione
Anno


LUOGOLAVORO
Cognome
IMPIEGATO
LUOGONASCITA
CITTÀ
Nome
Nome
Età
NumeroAbitanti
DataNascita

Esempio


DIREZIONE
Codice
Cognome
Stipendio
Età
IMPIEGATO
APPARTENENZA
DataInizio
DIPARTIMENTO
Telefono
Nome
PARTECIPAZIONE
DataInizio
COMPOSIZIONE
Nome
Budget
DataConsegna
PROGETTO
FILIALE
Città
Ind.
Via
Numero civico
Codice postale

Cardinalità delle relazioni

- Data una relazione è possibile specificare una cardinalità minima ed una massima per ognuna delle entità coinvolte.
- La cardinalità minima specifica il minimo numero di occorrenze della relazione a cui deve partecipare ogni istanza dell'entità considerata.
- Viceversa, la cardinalità massima specifica il massimo numero di occorrenze della relazione a cui può partecipare ogni istanza dell'entità considerata.
- Esempio:

Cardinalità delle relazioni

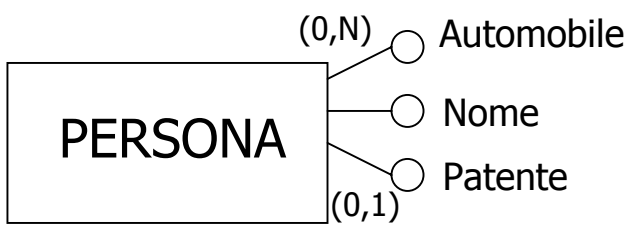
- In generale si può specificare qualsiasi numero intero non negativo come cardinalità.
- L'unico vincolo da rispettare è che la cardinalità minima sia minore od uguale alla cardinalità massima.
- Tuttavia, nella maggior parte dei casi sono sufficienti tre valori: zero, uno o N (detto anche "many", i.e., "molti") che indica un intero generico maggiore di uno.
- Per la cardinalità minima si ha quanto segue:
  - Se vale zero, si dice che la partecipazione alla relazione dell'entità relativa è opzionale.
  - Se vale uno, si dice che la partecipazione alla relazione dell'entità relativa è obbligatoria.
- Per la cardinalità massima si ha quanto segue:
  - Se vale uno, la partecipazione alla relazione dell'entità relativa può essere vista come una funzione (al più un'occorrenza della relazione può essere associata ad ogni occorrenza dell'entità).
  - Se vale N, ogni occorrenza dell'entità relativa è associata con un numero arbitrario di occorrenze della relazione.

Cardinalità delle relazioni

- Nel caso delle relazioni binarie (i.e., relazioni che coinvolgono due entità), si ha la seguente classificazione, ottenuta osservando le cardinalità massime:
  - Se entrambe le entità partecipano con cardinalità massima uguale a uno, la relazione è detta *uno-a-uno (one-to-one)*.
  - Se un'entità partecipa con cardinalità massima uguale a uno e l'altra con cardinalità massima uguale a N, la relazione è detta *uno-a-molti (one-to-many)*.
  - Se entrambe le entità partecipano con cardinalità massima uguale a N, la relazione è detta *molti-a-molti (many-to-many)*.

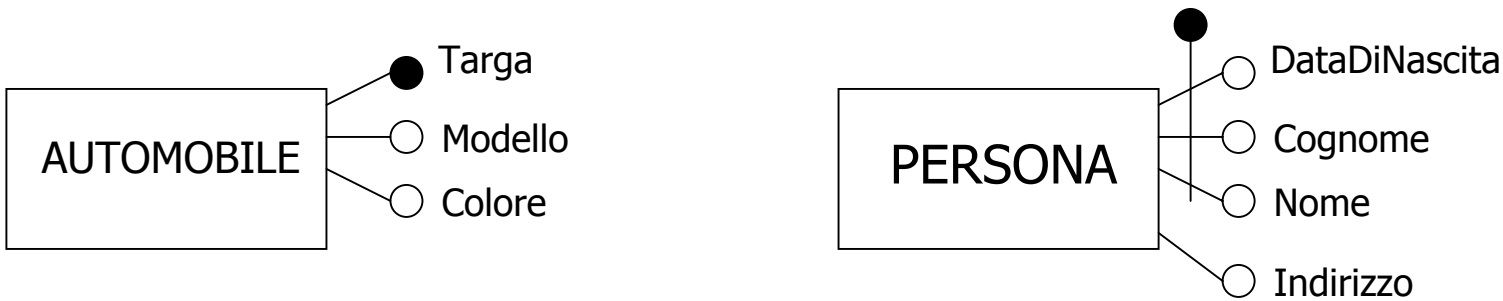
Cardinalità degli attributi

- Le cardinalità di un attributo specificano il minimo ed il massimo numero di valori dell'attributo che possono essere associati ad un'occorrenza dell'entità a cui si riferiscono.
- Se la cardinalità minima di un attributo è zero, esso viene detto opzionale.
- Se la cardinalità minima di un attributo è uno, esso viene detto obbligatorio.
- Se la cardinalità massima di un attributo è N, esso viene detto multivalore.
- Quando la cardinalità di un attributo è (1,1) viene omessa.
- Per quanto riguarda gli attributi multivalore, è opportuno notare che la situazione che rappresentano può essere modellata introducendo nuove entità e delle relazioni uno-a-molti o molti-a-molti.



Identificatori

- Ad ogni entità è possibile associare degli identificatori che permettono di distinguere senza ambiguità ogni singola occorrenza dell'entità stessa.
- Se l'identificatore è composto da uno o più attributi dell'entità a cui si riferisce, è detto *interno (o chiave)*.

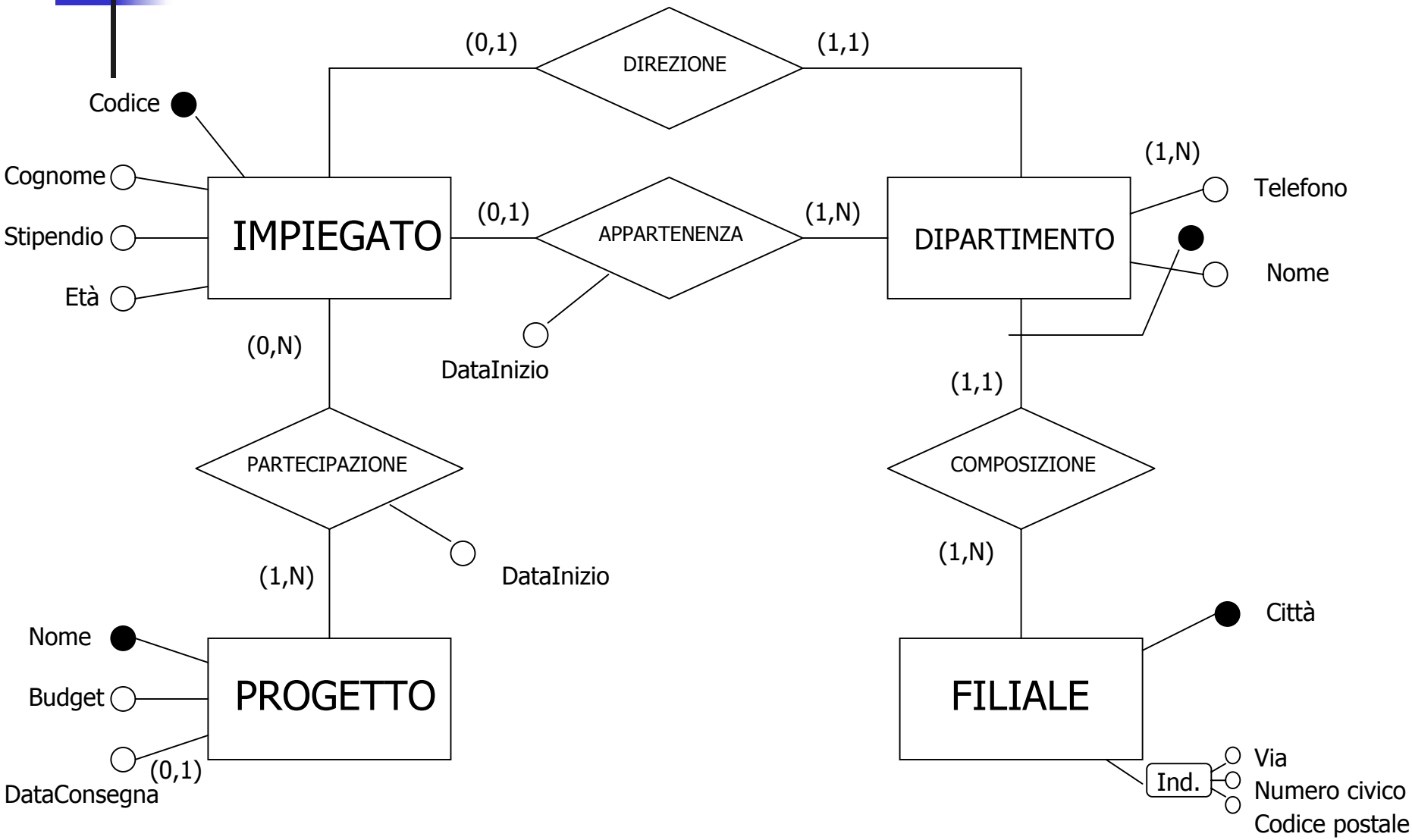


- Quando gli attributi di un'entità non permettono di identificare in modo univoco le sue occorrenze, si può ricorrere ad un attributo di un'altra entità collegata tramite una relazione uno-a-molti (identificatore esterno).

Identificatori

- Per quanto riguarda gli identificatori vale quanto segue:
  - Ogni identificatore può essere composto da uno o più attributi, se ognuno di questi ha cardinalità (1,1).
  - Un identificatore esterno può coinvolgere varie entità, se l'entità per cui viene definito partecipa con cardinalità (1,1) ad ogni relazione che coinvolge le entità esterne.
  - Un identificatore esterno può coinvolgere un'entità a sua volta identificata esternamente, a patto che non si generino dei cicli.
  - Ogni entità deve avere almeno un identificatore. Se ne ha più di uno, allora gli attributi e le entità coinvolte possono essere opzionali.

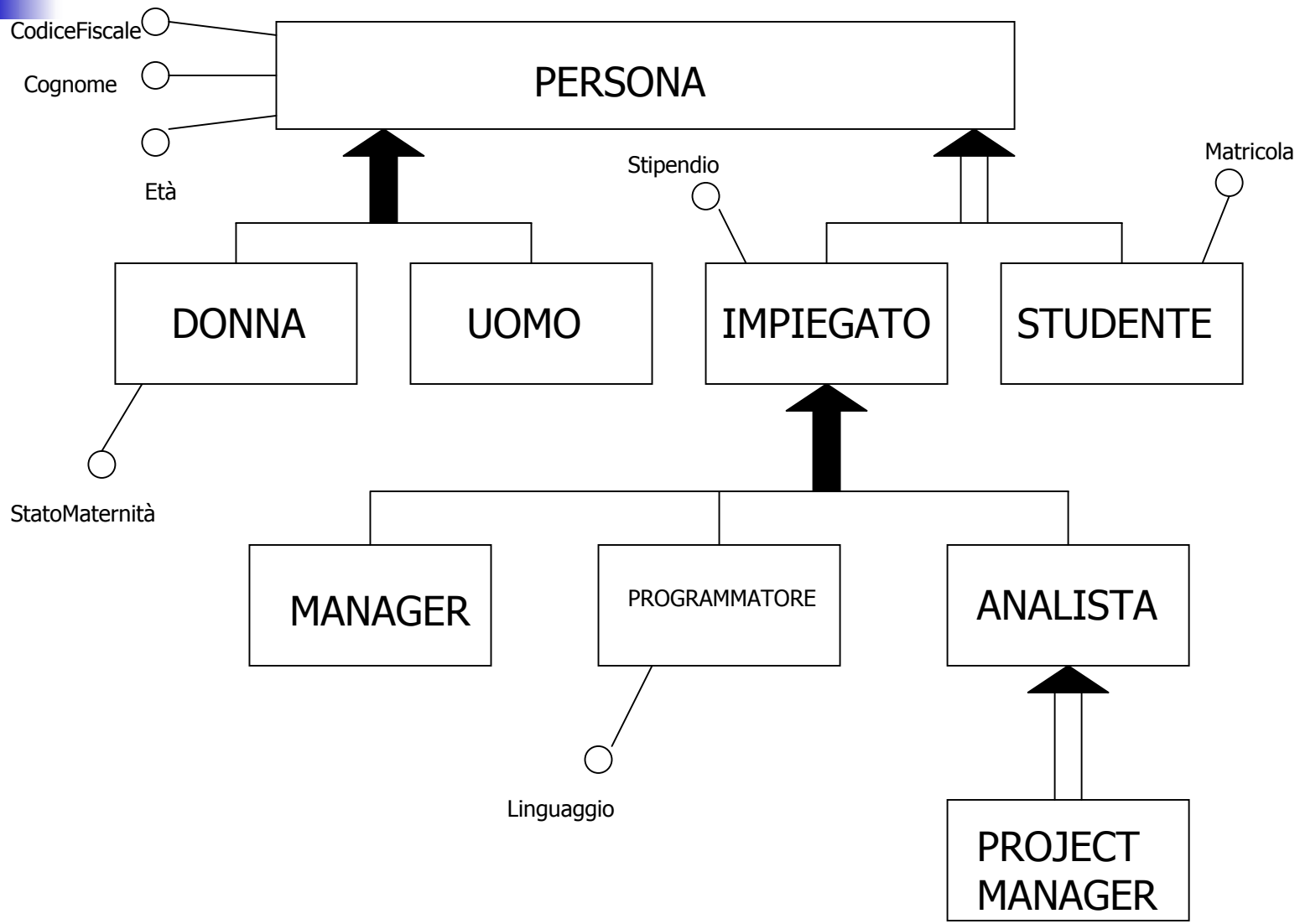
Esempio


(0,1)
DIREZIONE
(1,1)
Codice
Cognome
Stipendio
Età
IMPIEGATO
(0,1)
APPARTENENZA
(1,N)
DIPARTIMENTO
(1,N)
Telefono
Nome
DataInizio
(0,N)
(1,1)
PARTECIPAZIONE
COMPOSIZIONE
DataInizio
(1,N)
(1,N)
Città
Nome
Budget
PROGETTO
FILIALE
(0,1)
DataConsegna
Ind.
Via
Numero civico
Codice postale

Generalizzazioni

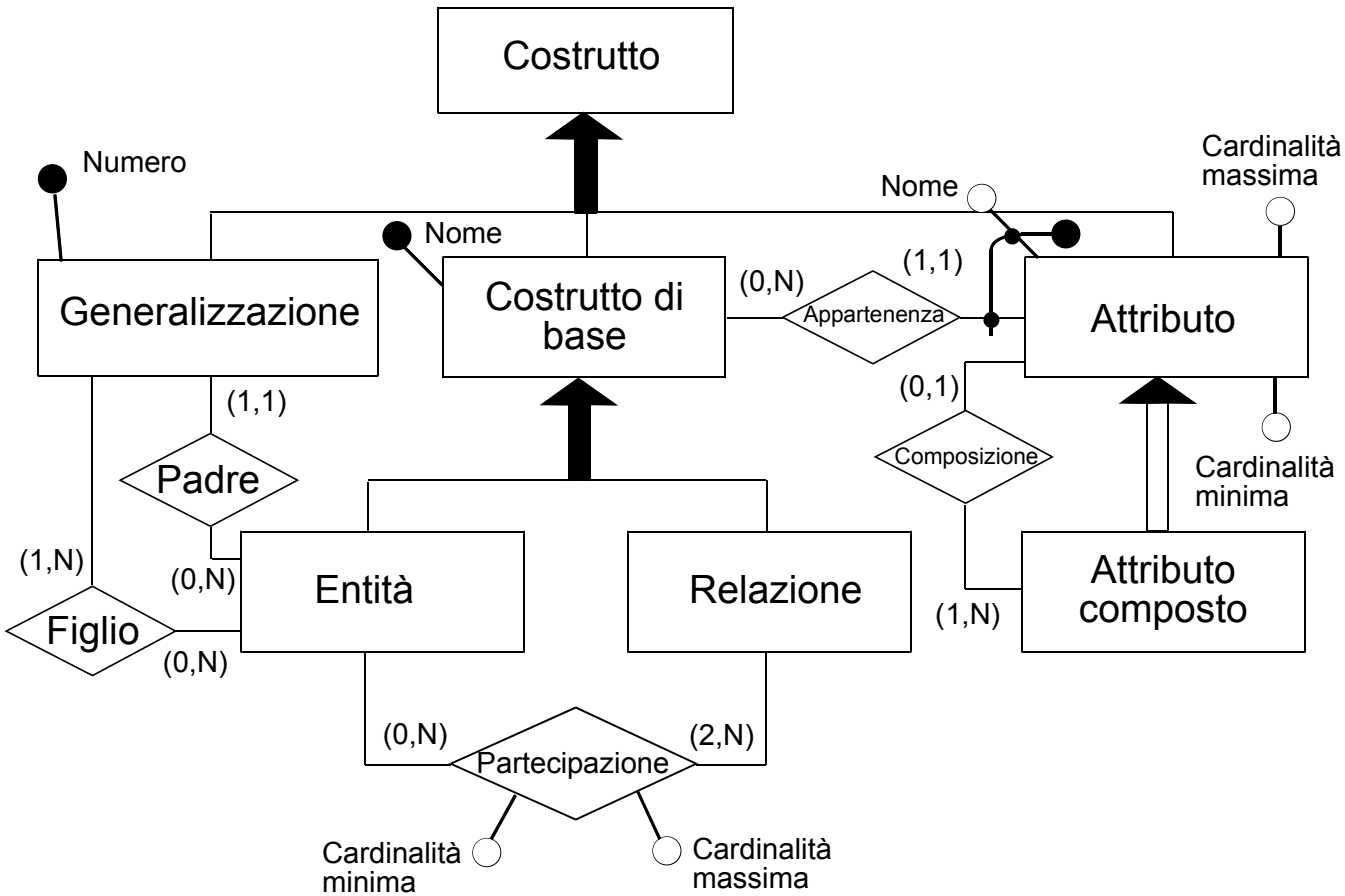
- Una generalizzazione lega un'entità padre E con delle entità figlie $E_1,\ldots,E_n$.
- E è detta generalizzazione di $E_1,\ldots,E_n$, mentre queste ultime sono dette specializzazioni di E.
- Tre le entità coinvolte in una generalizzazione valgono le seguenti proprietà:
  - Ogni occorrenza di un'entità figlia è anche un'occorrenza dell'entità padre.
  - Ogni proprietà (e.g., attributi identificatori ecc.) dell'entità padre è anche una proprietà delle entità figlie. Tale fatto viene detto *ereditarietà*.
- Una generalizzazione è detta *totale* se ogni occorrenza dell'entità padre è anche un'occorrenza di una delle entità figlie, altrimenti è detta *parziale*.
- Una generalizzazione è detta *esclusiva* se ogni occorrenza dell'entità padre è al più un'occorrenza di una delle entità figlie, altrimenti è detta *sovrapposta*.
- Quando una generalizzazione ha una sola entità figlia si parla di *sottoinsieme*.

Generalizzazioni



Riepilogo del modello E-R

- Il modello E-R è talmente generale da poter essere usato per descrivere se stesso:



Raccolta ed analisi dei requisiti

- Per raccolta dei requisiti di un'applicazione si intende:
  - L'individuazione dei problemi che l'applicazione deve risolvere.
  - La definizione delle caratteristiche dell'applicazione:
    - Aspetti statici (dati).
    - Aspetti dinamici (operazioni sui dati).
- La raccolta dei requisiti avviene generalmente in linguaggio naturale.
- Per analisi dei requisiti si intende invece la razionalizzazione e l'organizzazione dei requisiti raccolti.
- Le due attività sono fortemente connesse alimentandosi a vicenda.

Fonti di informazione

- La raccolta dei requisiti avviene attingendo da varie fonti di informazione:
  - Utenti dell'applicazione
  - Documentazione esistente
  - Eventuali versioni preesistenti dell'applicazione
- Nella fase di raccolta dei requisiti è fondamentale l'interazione con gli utenti del sistema informativo al fine di individuare gli aspetti essenziali, isolando i problemi marginali a raffinamenti e studi successivi.
- Siccome viene usato il linguaggio naturale, è importante analizzare profondamente i testi raccolti in modo da eliminare eventuali ambiguità ed inesattezze.

Esempio: società di formazione

- Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei **partecipanti** ai corsi e dei **docenti**. Per i **partecipanti** (circa 5000), identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, il **luogo** di nascita, il nome dei loro attuali datori di lavoro, i **posti** dove hanno lavorato in precedenza insieme al periodo, l'indirizzo ed il **numero di telefono**, i corsi che hanno frequentato (i corsi sono circa 200) e il **giudizio** finale. Rappresentiamo anche i **seminari** che stanno frequentando e, per ogni giorno, i **luoghi** e le ore dove sono tenute le lezioni. I corsi hanno un codice, un titolo e possono avere varie **edizioni** con date di inizio e fine e numero di partecipanti. Se gli **studenti** sono liberi professionisti, vogliamo conoscere l'area di interesse e, se lo possiedono, il **titolo**. **Per quelli che lavorano alle dipendenze di altri**, vogliamo conoscere invece il loro livello e la posizione ricoperta. Per gli **insegnanti** (circa 300) rappresentiamo il cognome, l'età, il posto dove sono nati, il nome del corso che insegnano, quelli che hanno insegnato nel passato e quelli che possono insegnare. Rappresentiamo anche tutti i loro **recapiti telefonici**. I **docenti** possono essere dipendenti interni della società o collaboratori esterni.

Regole generali

- Scegliere il livello giusto di astrazione: nell'esempio precedente "titolo" andrebbe specificato meglio come "titolo professionale" e "giudizio" come "votazione in decimi".
- Utilizzare una struttura standard delle frasi: lo stile deve essere uniforme; ad esempio "per <dato> rappresentiamo <insieme di proprietà>".
- Evitare frasi contorte: al posto di "a quelli che lavorano alle dipendenze di altri" è preferibile usare "lavoratori dipendenti".
- Individuare i sinonimi/omonimi e unificare i termini: nell'esempio precedente si usa sia il termine "docente" che il termine "insegnante". Inoltre "posto" è usato sia riferito a impiego, che a città, che ad aula.
- Esplicitare il riferimento tra termini: nell'esempio non è chiaro se "indirizzo" e "numero di telefono" si riferiscano ai partecipanti o ai loro datori di lavoro. Inoltre non è chiaro se l'espressione "Per quelli che lavorano" si rivolga ai partecipanti o ai docenti.
- Costruire un glossario dei termini: la definizione di un glossario con una descrizione, sinonimi e termini correlati logicamente per ogni termine usato nei requisiti può essere molto utile per agevolare la comprensione.

Esempio: società di formazione

- Glossario:

Termine	**Descrizione**	**Sinonimi**	**Collegamenti**
Partecipante	Partecipante ai corsi. Può essere un dipendente o un professionista.	Studente	Corso, Datore
Docente	Docente dei corsi. Può essere un collaboratore esterno	Insegnante	Corso
Corso	Corso offerto. Può avere varie edizioni.	Seminario	Docente, Partecipante
Datore	Datori di lavoro attuali e passati dei partecipanti ai corsi.	Posto	Partecipante

Esempio: società di formazione

- Frasi di carattere generale:
  - Si vuole realizzare una base di dati per una società che eroga corsi, di cui vogliamo rappresentare i dati dei partecipanti ai corsi e dei docenti.
- Frasi relative ai partecipanti:
  - Per i partecipanti (circa 5000), identificati da un codice, si vuole memorizzare il codice fiscale, il cognome, l'età, il sesso, la città di nascita, il nome dei loro attuali datori di lavoro e di quelli precedenti (insieme alle date di inizio e fine rapporto), le edizioni dei corsi che stanno attualmente frequentando e quelli che hanno frequentato nel passato, con la relativa votazione in decimi.
- Frasi relative ai datori di lavoro:
  - Relativamente ai datori di lavoro presenti e passati dei partecipanti, rappresentiamo il nome, l'indirizzo e il numero di telefono.

Esempio: società di formazione

- Frasi relative ai corsi:
  - Per i corsi (circa 200), rappresentiamo il titolo e il codice, le varie edizioni con date di inizio e fine e, per ogni edizione, rappresentiamo il numero di partecipanti e il giorno della settimana, le aule e le ore dove sono tenute le lezioni.
- Frasi relative ai tipi specifici di partecipanti:
  - Per i partecipanti che sono liberi professionisti, rappresentiamo l'are di interesse e, se lo possiedono, il titolo professionale. Per i partecipanti che sono dipendenti, rappresentiamo invece il loro livello e la posizione ricoperta.
- Frasi relative ai docenti:
  - Per i docenti (circa 300), rappresentiamo il cognome, l'età, la città di nascita, tutti i numeri di telefono, il titolo del corso che insegnano, di quelli che hanno insegnato nel passato e di quelli che possono insegnare. I docenti possono essere dipendenti interni della società di formazione o collaboratori esterni.

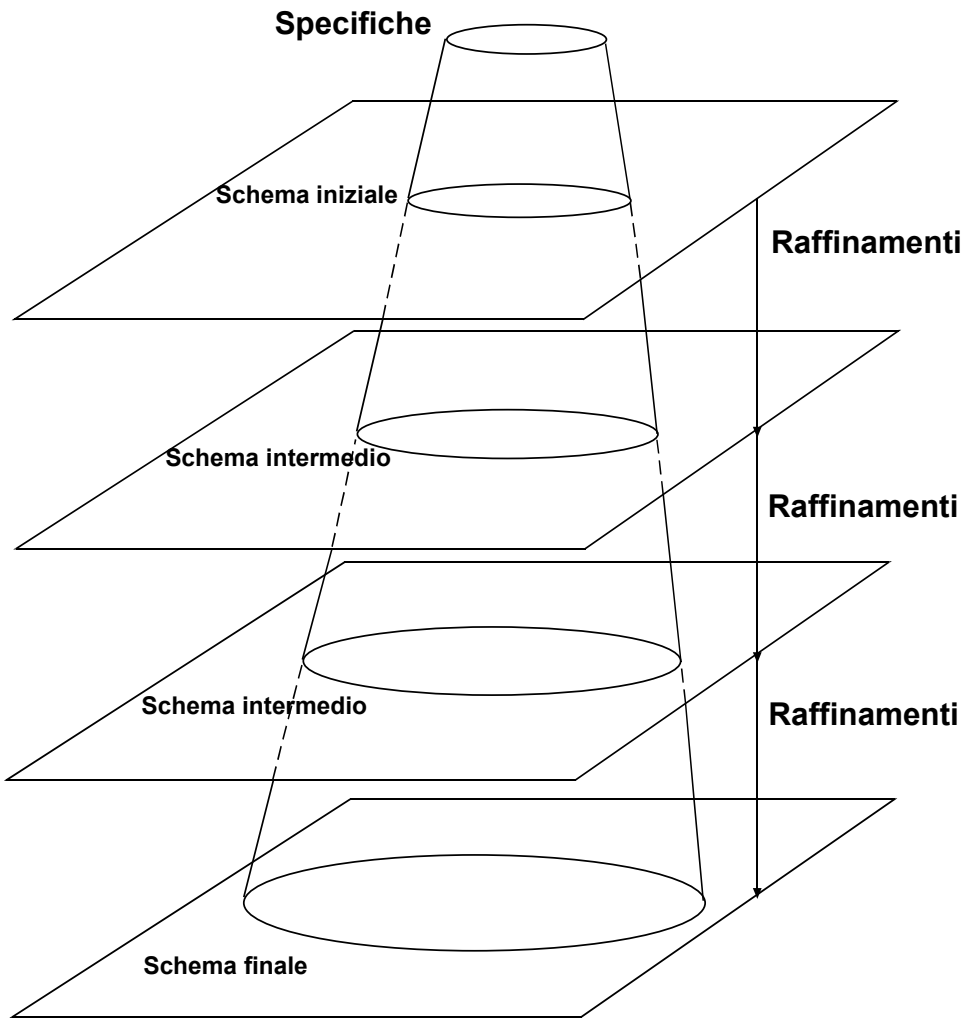
Esempio: società di formazione

- Operazione 1: inserimento di un nuovo partecipante con tutti i suoi dati (in media 40 volte al giorno).
- Operazione 2: assegnamento di un partecipante ad una edizione di un corso (50 volte al giorno).
- Operazione 3: inserimento di un nuovo docente con tutti i suoi dati ed i corsi che può insegnare (2 volte al giorno).
- Operazione 4: assegnamento di un docente qualificato ad una edizione di un corso (15 volte al giorno).
- Operazione 5: stampa di tutte le informazioni sulle edizioni passate di un corso con titolo, orari lezioni e numero di partecipanti (10 volte al giorno).
- Operazione 6: stampa di tutti i corsi offerti, con informazioni sui docenti abilitati (20 volte al giorno).
- Operazione 7: per ogni docente, ritrovamento di tutti i partecipanti a tutti i corsi da lui insegnati (5 volte alla settimana).
- Operazione 8: statistica su tutti i partecipanti ad un corso con tutte le informazioni su di essi, sull'edizione alla quale hanno partecipato e la relativa votazione (10 volte al mese).

La strategia top-down

- La strategia top-down consente di produrre lo schema concettuale attraverso una fase di raffinamenti successivi a partire da uno schema iniziale che riassume tutte le specifiche in modo molto astratto.
- Ad ogni raffinamento si applica una trasformazione che aumenta il grado di dettaglio dei vari concetti presenti nello schema.
- Quindi, ad ogni passo, la strategia top-down contiene tutti gli aspetti delle specifiche.
- Il vantaggio per il progettista è quello di mantenere una visione globale sull'intero sistema per entrare nei dettagli solamente al momento opportuno.
- Lo svantaggio principale è che raramente nelle applicazioni di una certa complessità si ha subito una visione globale.

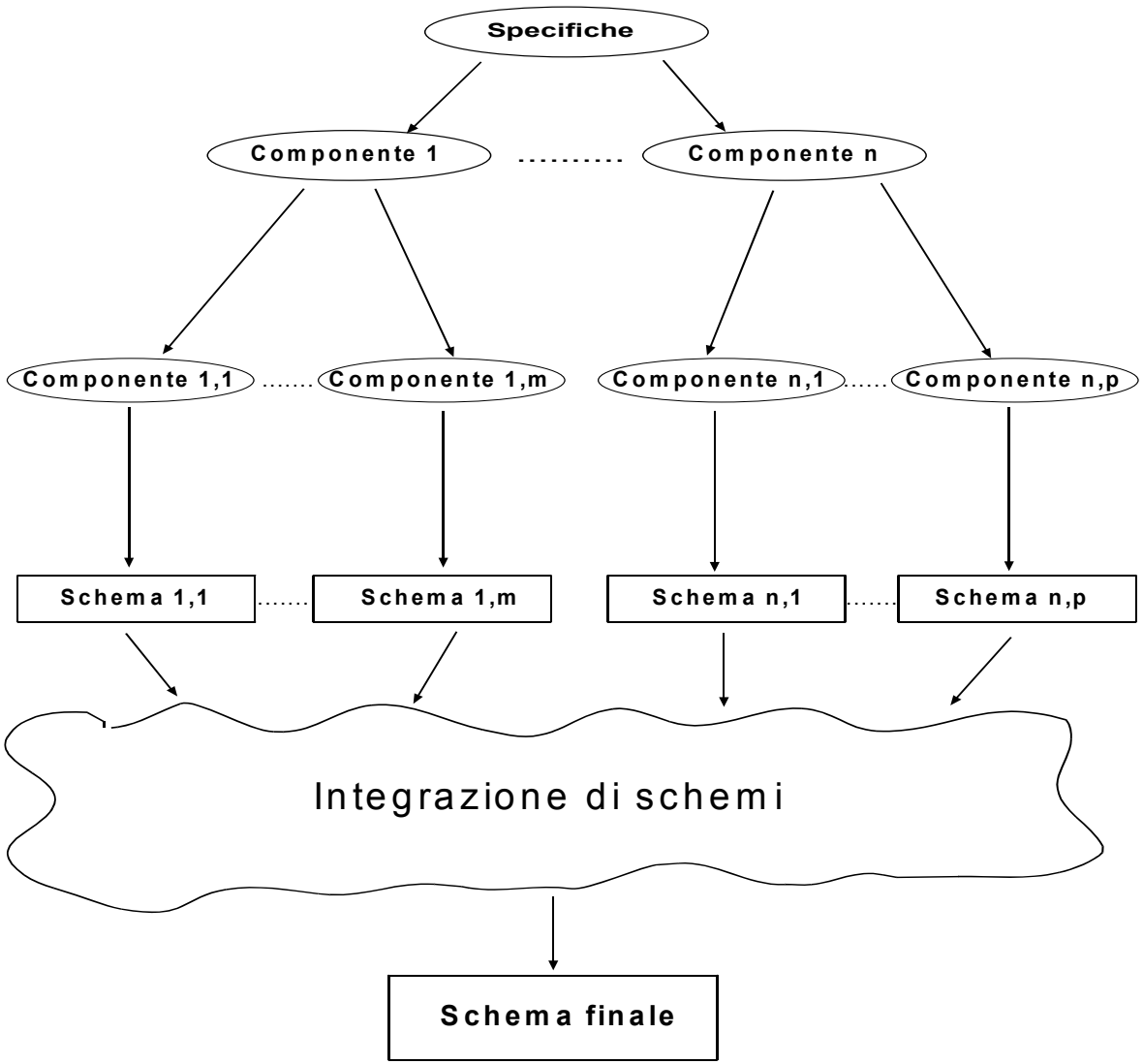
La strategia top-down



La strategia bottom-up

- La strategia bottom-up consente di realizzare lo schema concettuale attraverso una suddivisione progressiva delle specifiche iniziali in componenti sempre più piccole fino ad arrivare a dei frammenti elementari.
- Successivamente ogni frammento elementare viene rappresentato per mezzo di un semplice schema concettuale.
- Infine tutti i singoli schemi vengono progressivamente fusi fino ad ottenere lo schema concettuale generale.
- Il vantaggio è che il problema viene ridotto ad una serie di problemi elementari facilmente individuabili. Ognuno di questi può essere seguito da un progettista distinto; quindi si tratta di una strategia che ben si presta al lavoro cooperativo.
- Lo svantaggio principale è che l'integrazione di schemi concettuali distinti presenta spesso delle difficoltà notevoli in applicazioni complesse.

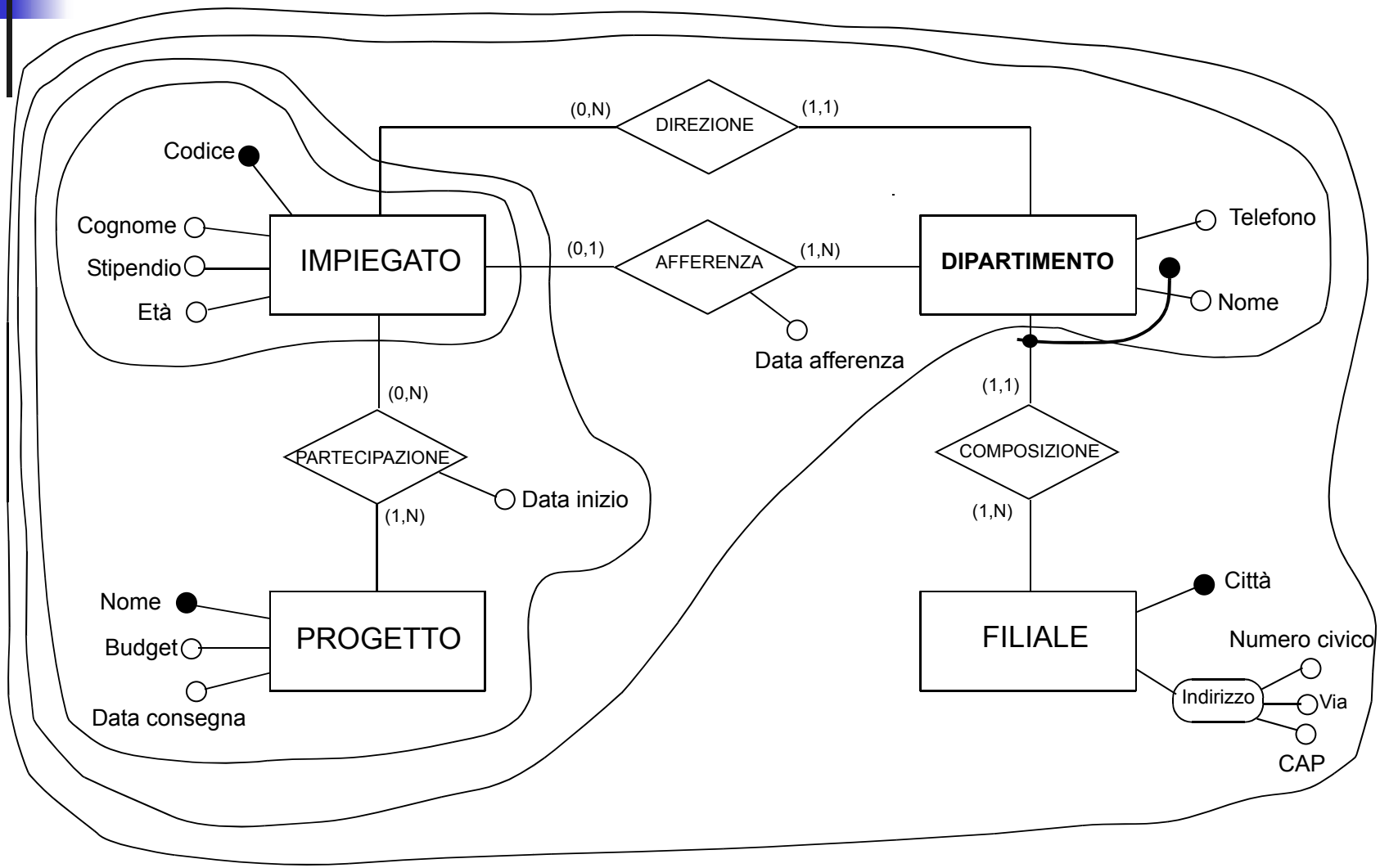
La strategia bottom-up



Strategia inside-out

- La strategia inside-out è una variante di quella bottom-up.
- Si inizia individuando soltanto alcuni concetti fondamentali nelle specifiche, procedendo in seguito a "macchia d'olio", i.e., si rappresentano in primis i concetti più strettamente legati a quelli iniziali e poi via via gli altri, "navigando" tra le specifiche.
- Il vantaggio della strategia è che non richiede passi di integrazione fra schemi distinti.
- Lo svantaggio è che di volta in volta bisogna analizzare le specifiche nel loro complesso per individuare i concetti ancora mancanti nello schema e descriverli in dettaglio.

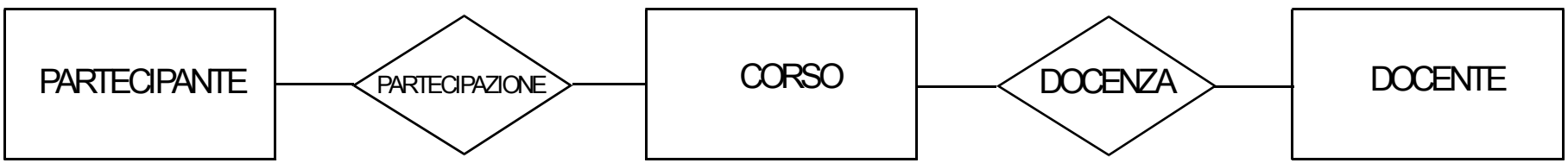
Strategia inside-out



La strategia mista

- La strategia mista cerca di combinare i vantaggi di quella top-down con quelli della strategia bottom-up.
- Si inizia suddividendo i requisiti in componenti separate, di cui le principali vengono rappresentate a livello astratto in uno schema scheletro.
- Lo schema scheletro favorisce l'integrazione dei vari sottoschemi che vengono prodotti separatamente.
- Infatti, a partire dallo schema scheletro, è possibile procedere in modalità top-down (attraverso raffinamenti successivi), bottom-up (integrando lo schema con concetti non ancora rappresentati) o inside-out (procedendo a macchia d'olio).
- Data la sua versatilità, la strategia mista è quella che meglio si adatta alle varie situazioni, dove le altre strategie falliscono.

La strategia mista



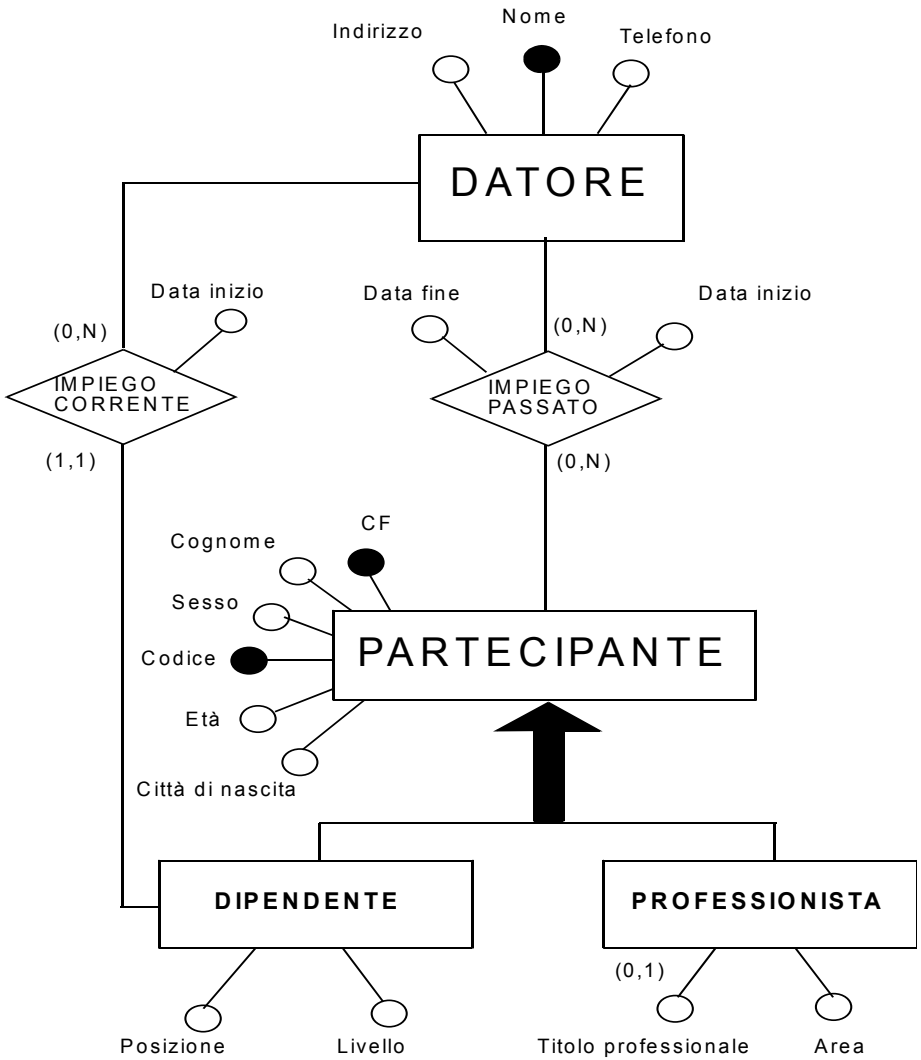
Qualità di uno schema

- Ogni schema concettuale deve garantire alcune proprietà generali che ne garantiscano la qualità:
  - Correttezza: i costrutti del modello usato sono utilizzati in modo appropriato, i.e., non ci sono errori sintattici (e.g., generalizzazioni fra relazioni invece che tra entità) o semantici (uso di una relazione per descrivere il fatto che un'entità è la specializzazione di un'altra entità).
  - Completezza: tutti i dati di interesse devono essere rappresentati e tutte le operazioni devono poter essere eseguite utilizzando i concetti descritti nello schema.
  - Leggibilità: lo schema deve essere autoesplicativo; quindi vanno usati nomi significativi per i vari costrutti e vanno seguite alcune regole pratiche:
    - I costrutti vanno disposti in modo uniforme, mettendo al centro quelli che partecipano a più relazioni.
    - Le linee tracciate devono essere perpendicolari e bisogna cercare di minimizzarne le intersezioni.
    - Le entità padri vanno poste sopra alle entità figlie (nel caso delle generalizzazioni).
  - Minimalità: tutte le specifiche dovrebbero essere rappresentate una sola volta nello schema, i.e., non ci devono essere ridondanze.

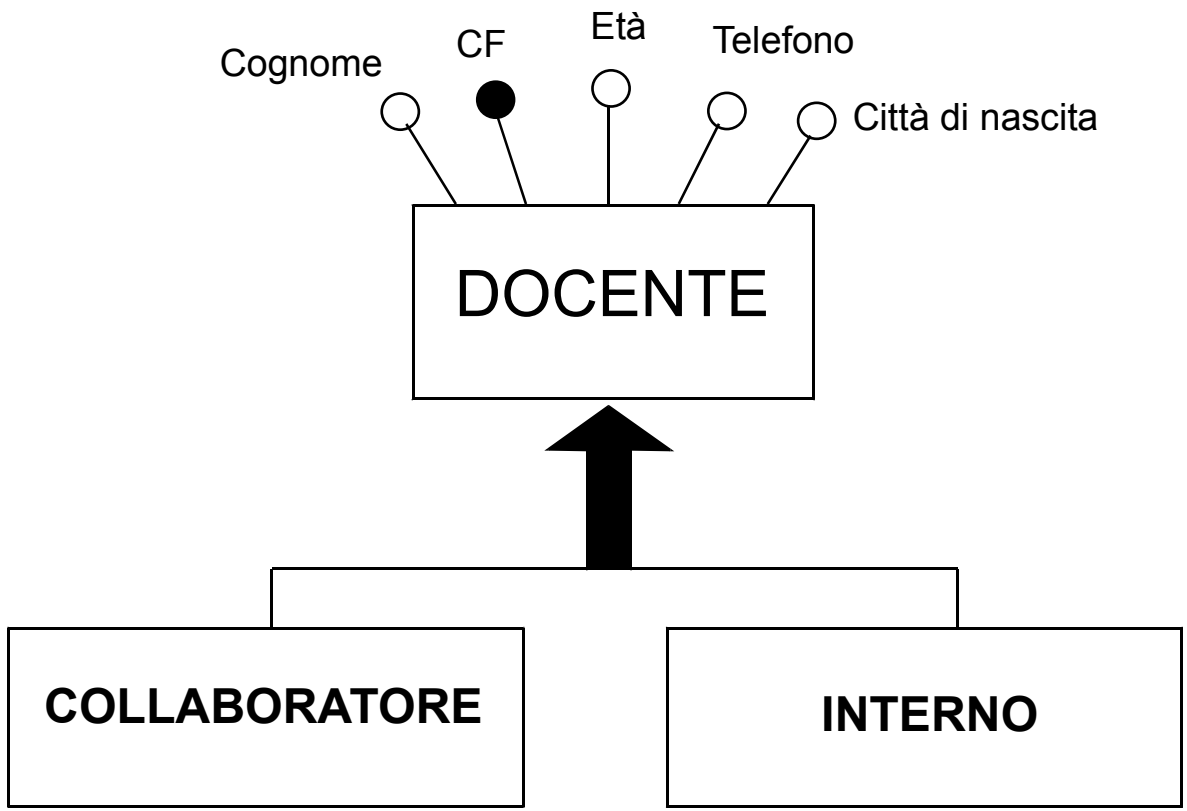
Metodologia generale

- Analisi dei requisiti:
  - Costruzione di un glossario dei termini
  - Analisi dei requisiti ed eliminazione delle eventuali ambiguità
  - Raggruppamento dei requisiti in insiemi omogenei
- Passo base:
  - Individuazione dei concetti maggiormente rilevanti e loro rappresentazione in uno schema scheletro.
- Passo di decomposizione (opzionale):
  - Decomposizione dei requisiti relativamente ai concetti presenti nello schema scheletro.
- Passo iterativo (da ripetere per ogni sottoschema fino alla rappresentazione di tutte le specifiche):
  - Raffinamento dei concetti presenti in base alle loro specifiche.
  - Aggiunta di nuovi concetti allo schema per descrivere nuove cose.
- Passo di integrazione:
  - Integrazione in un unico schema dei vari sottoschemi, con riferimento allo schema scheletro.
- Analisi di qualità:
  - Verifica della correttezza, completezza, minimalità e leggibilità dello schema (con eventuale ristrutturazione di quest'ultimo).

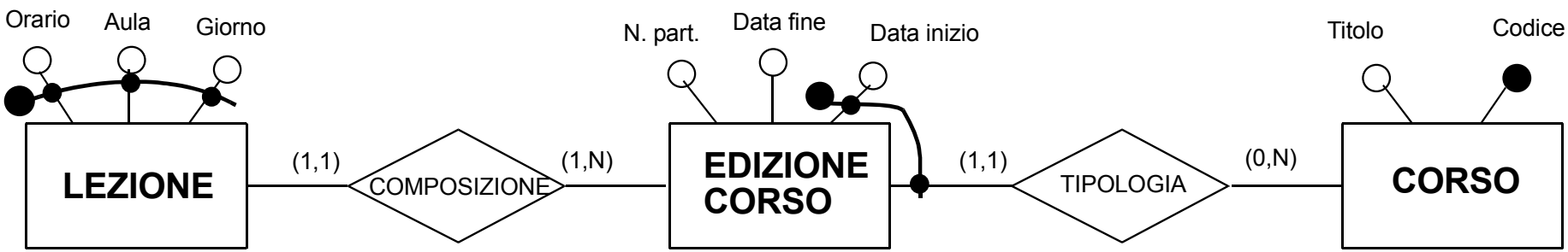
Esempio: società di formazione



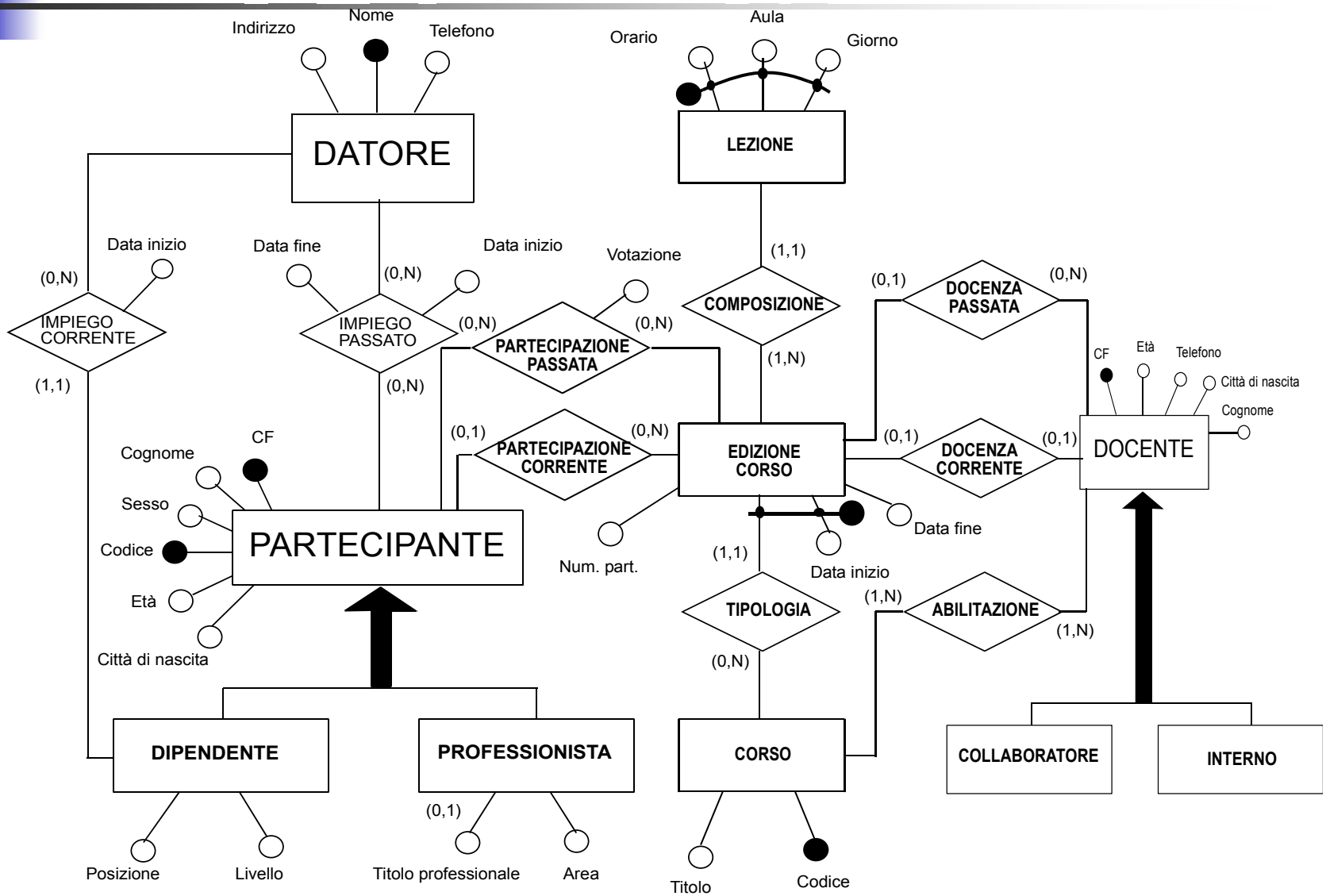
Esempio: società di formazione



Esempio: società di formazione



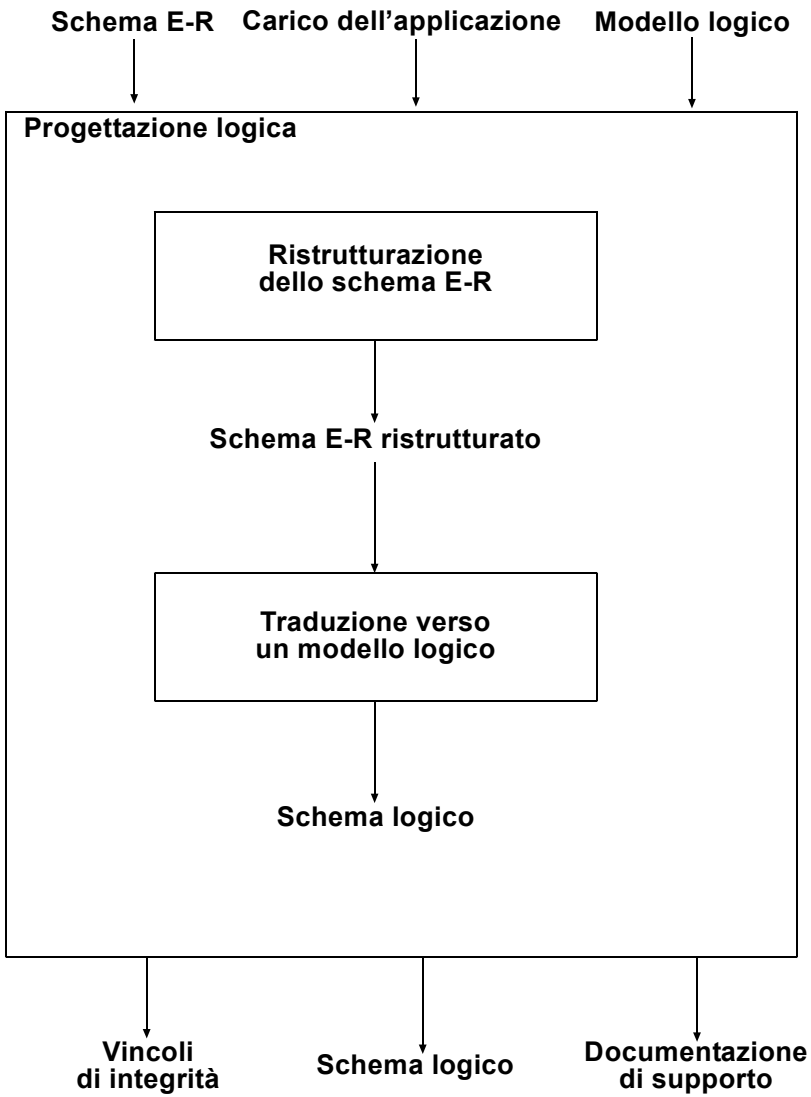
Esempio: società di formazione



Progettazione logica

- Lo scopo della fase di progettazione logica è di produrre uno schema logico, in accordo al modello dei dati scelto, che rappresenti fedelmente l'informazione descritta da uno schema E-R (prodotto nella fase di progettazione concettuale).
- Nel seguito assumiamo come modello dei dati il modello relazionale.
- Una procedura di traduzione diretta dallo schema E-R a quello logico corrispondente non è possibile per le seguenti ragioni:
  - Vi sono costrutti del modello E-R che non hanno un costrutto naturalmente corrispondente nel modello relazionale (e.g., le generalizzazioni).
  - Mentre uno schema concettuale è ad alto livello ed indipendente dalla tecnologia scelta per l'implementazione, lo schema logico è il punto di partenza di quest'ultimo e quindi deve tener conto del fattore prestazioni del prodotto finale.
- Quindi la fase di progettazione logica si divide in due parti:
  - Ristrutturazione dello schema E-R in base a criteri di ottimizzazione e di semplificazione.
  - Traduzione dello schema ristrutturato in uno schema logico ed eventuale ulteriore ottimizzazione.

Progettazione logica



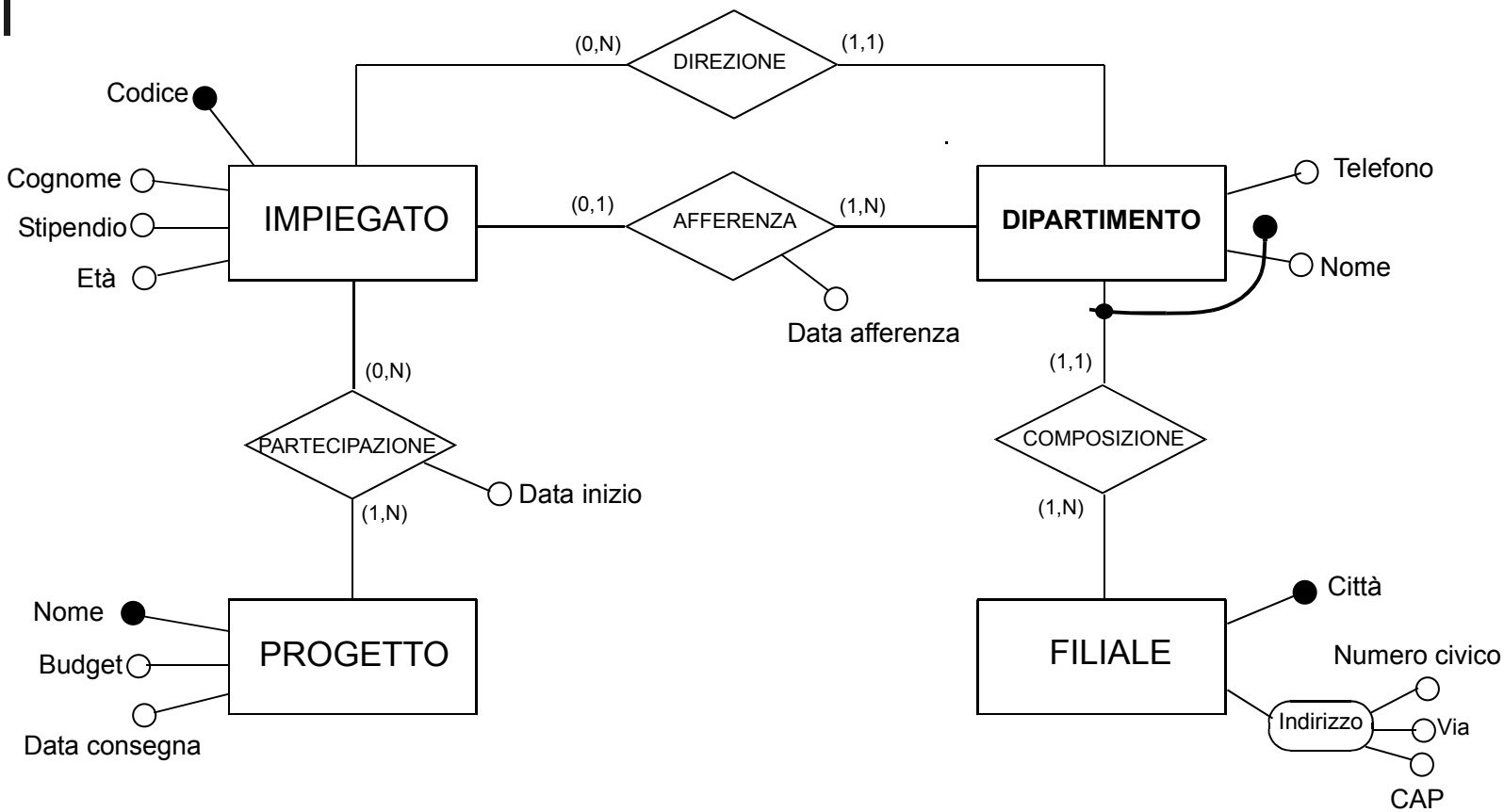
Analisi delle prestazioni

- Scopo di questa fase è di effettuare uno studio di massima per ottimizzare degli indici di prestazione del sistema.
- Si preferisce parlare di indici di prestazione, piuttosto che di prestazioni, in quanto queste ultime dipendono da parametri fisici (e.g., il particolare DBMS che verrà adottato) e quindi sono difficilmente stimabili in fase di progettazione logica.
- In linea di massima i parametri che influiscono maggiormente sulle prestazioni sono:
  - Costo delle operazioni (valutato nel numero di occorrenze di entità e associazioni che vanno "visitate" per portare a termine le varie operazioni).
  - Occupazione di memoria (valutato in termini dello spazio necessario per memorizzare i dati descritti dallo schema E-R).
- Per l'analisi servono, oltre allo schema E-R, le seguenti informazioni:
  - Volume dei dati: numero di occorrenze di ogni entità e associazione, dimensioni degli attributi.
  - Caratteristiche delle operazioni: tipo (interattiva o batch), frequenza, dati coinvolti (entità e/o associazioni).
- In pratica vale la *regola "ottanta-venti"*, i.e., l'80% del carico è generato dal 20% delle operazioni.

Analisi delle prestazioni

- Il volume dei dati può essere rappresentato tramite una tabella riportante in ogni riga il concetto considerato, il tipo (entità o associazione) ed il volume (i.e., il numero di occorrenze) previsto a pieno regime.
- Per quanto riguarda le associazioni, il numero delle occorrenze dipende da due parametri:
  - Il numero di occorrenze delle entità coinvolte nell'associazione.
  - Il numero medio di partecipazioni di un'occorrenza di entità alle occorrenze di associazioni (in pratica questo parametro dipende dalla cardinalità delle associazioni stesse).
- Le operazioni possono essere rappresentate in un'altra tabella in cui in ogni riga si riporta l'operazione, il tipo e la frequenza.
- Ad ogni operazione inoltre è possibile associare uno schema di operazione: si tratta di un frammento dello schema E-R in cui compaiono soltanto i concetti coinvolti ed in cui viene indicato graficamente il cammino logico da percorrere per accedere alle informazioni necessarie per svolgere l'operazione.

Esempio


(0,N)
DIREZIONE
(1,1)
Codice
Cognome
Stipendio
Età
IMPIEGATO
(0,1)
AFFERENZA
(1,N)
DIPARTIMENTO
Telefono
Nome
Data afferenza
(0,N)
PARTECIPAZIONE
Data inizio
(1,N)
(1,1)
COMPOSIZIONE
(1,N)
Nome
Budget
Data consegna
PROGETTO
Città
FILIALE
Numero civico
Indirizzo
Via
CAP

Esempio

- Operazioni:
  - Operazione 1: assegna un impiegato ad un progetto.
  - Operazione 2: trova i dati di un impiegato, del suo dipartimento e dei progetti a cui sta lavorando.
  - Operazione 3: trova i dati di tutti gli impiegati di un dato dipartimento.
  - Operazione 4: per ogni sede trova i dipartimenti con il cognome del direttore e l'elenco degli impiegati che vi lavorano.

Esempio

Tabella dei volumi

Concetto	Tipo	Volume
Filiale	E	10
Dipartimento	E	80
Impiegato	E	2000
Progetto	E	500
Composizione	A	80
Afferenza	A	1900
Direzione	A	80
Partecipazione	A	6000

Tabella delle operazioni

Operazione	Tipo	Frequenza
Op. 1	I	50 al giorno
Op. 2	I	100 al giorno
Op. 3	I	10 al giorno
Op. 4	B	2 alla settimana

Esempio

Schema di Op. 2:

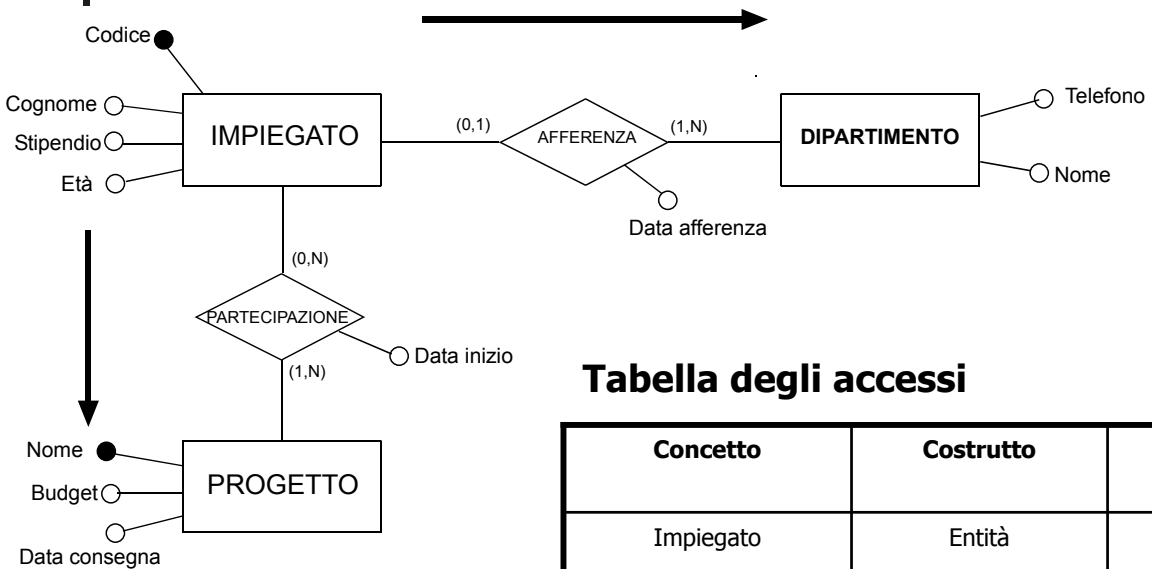


Tabella degli accessi

Concetto	Costrutto	Accessi	Tipo
Impiegato	Entità	1	L
Afferenza	Associazione	1	L
Dipartimento	Entità	1	L
Partecipazione	Associazione	3	L
Progetto	Entità	3	L

Ristrutturazione degli schemi E-R

- Per quanto riguarda la fase di ristrutturazione possiamo individuare i seguenti passi:
  - **Analisi delle ridondanze**: si deve decidere se mantenere o meno le ridondanze (i.e., i concetti derivabili da altri) presenti nello schema.
  - **Eliminazione delle generalizzazioni**: bisogna sostituire le generalizzazioni presenti nello schema con altri costrutti che siano naturalmente traducibili in costrutti del modello relazionale.
  - **Partizionamento/accorpamento di entità e associazioni**: a seconda dei casi, potrebbe essere utile suddividere ulteriormente dei concetti od accorparne altri strettamente correlati.
  - **Scelta degli identificatori primari**: per le entità che hanno più di un identificatore si deve indicare quello principale.

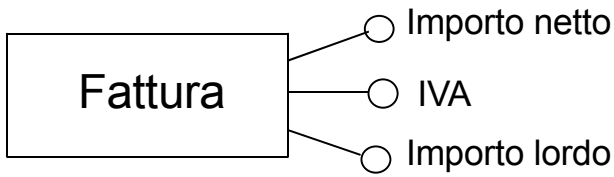
Analisi delle ridondanze

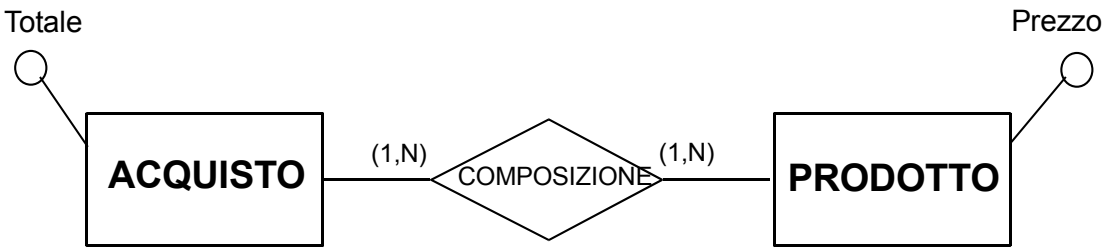
- Le ridondanze presentano il vantaggio di semplificare le interrogazioni, in quanto riducono il numero di accessi necessari per recuperare i dati necessari alle operazioni.
- Tuttavia vi sono dei lati negativi:
  - Le operazioni di aggiornamento risultano più onerose in presenza di concetti ridondanti (l'operazione va ripetuta per ogni occorrenza).
  - Vi è un maggior consumo di spazio (memoria).
- Quindi, di volta in volta, bisogna valutare i pro ed i contro in termini di guadagno di velocità delle operazioni sui dati e di consumo di memoria, prima di decidere se eliminare o mantenere una data ridondanza.

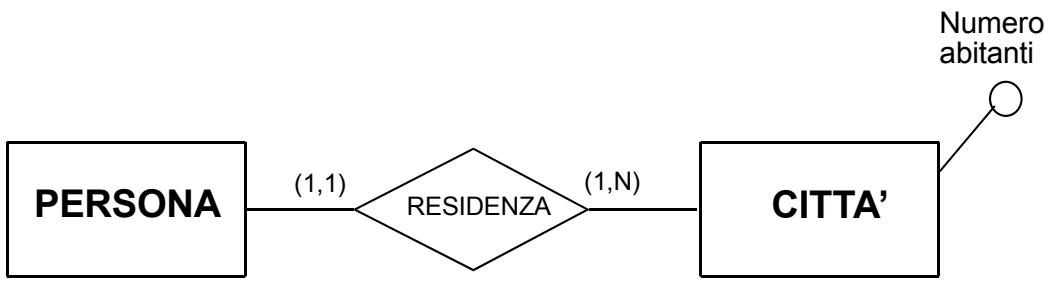
Tipologie di ridondanza

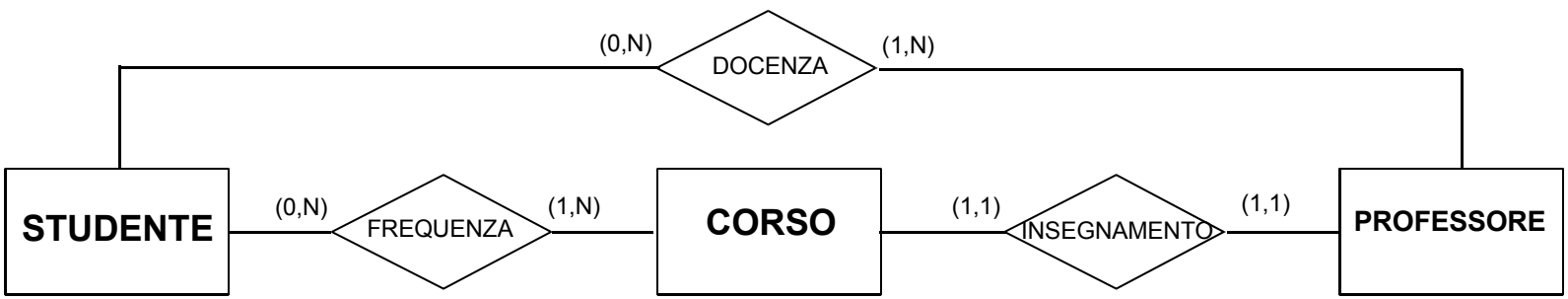
- I tipi di ridondanza che si possono avere in uno schema E-R sono i seguenti:
  - Attributi derivabili, occorrenza per occorrenza (di entità o associazione), da altri attributi dello stesso concetto.
  - Attributi derivabili da attributi di altre entità o associazioni (di solito rappresentano valori aggregati).
  - Attributi derivabili da operazioni di conteggio di occorrenze (è un caso particolare del precedente).
  - Associazioni derivabili dalla composizione di altre associazioni in presenza di cicli (si noti tuttavia che la semplice presenza di cicli in uno schema E-R non comporta necessariamente ridondanze di questo tipo).

Esempi di ridondanze









Esempio pratico

- Dato lo schema su persone e città del lucido precedente, consideriamo le seguenti operazioni:
  - Operazione 1: inserisci una nuova persona con la sua città di residenza.
  - Operazione 2: stampa tutti i dati di una città includendo anche il numero di abitanti.

Tabella dei volumi

Concetto	Tipo	Volume
Città	E	200
Persona	E	1000000
Residenza	A	1000000

Tabella delle operazioni

Operazione	Tipo	Frequenza
Op. 1	I	500 al giorno
Op. 2	I	2 al giorno

Esempio pratico

- Assumendo che siano richiesti 4 byte per memorizzare il numero di abitanti, in presenza dell'attributo ridondante abbiamo un utilizzo di memoria aggiuntivo pari a 4 x 200 = 800 byte.
- L'operazione 1 richiede:
  - 1 accesso in scrittura a PERSONA,
  - 1 accesso in scrittura a RESIDENZA,
  - 1 accesso in lettura ed 1 in scrittura a CITTA'
- L'operazione 2 richiede soltanto un accesso in lettura a CITTA'.
- Considerando trascurabile il costo dell'operazione 2 e contando doppio il costo degli accessi in scrittura (rispetto a quelli in lettura), abbiamo un costo totale di 3.500 accessi al giorno in presenza dell'attributo ridondante.

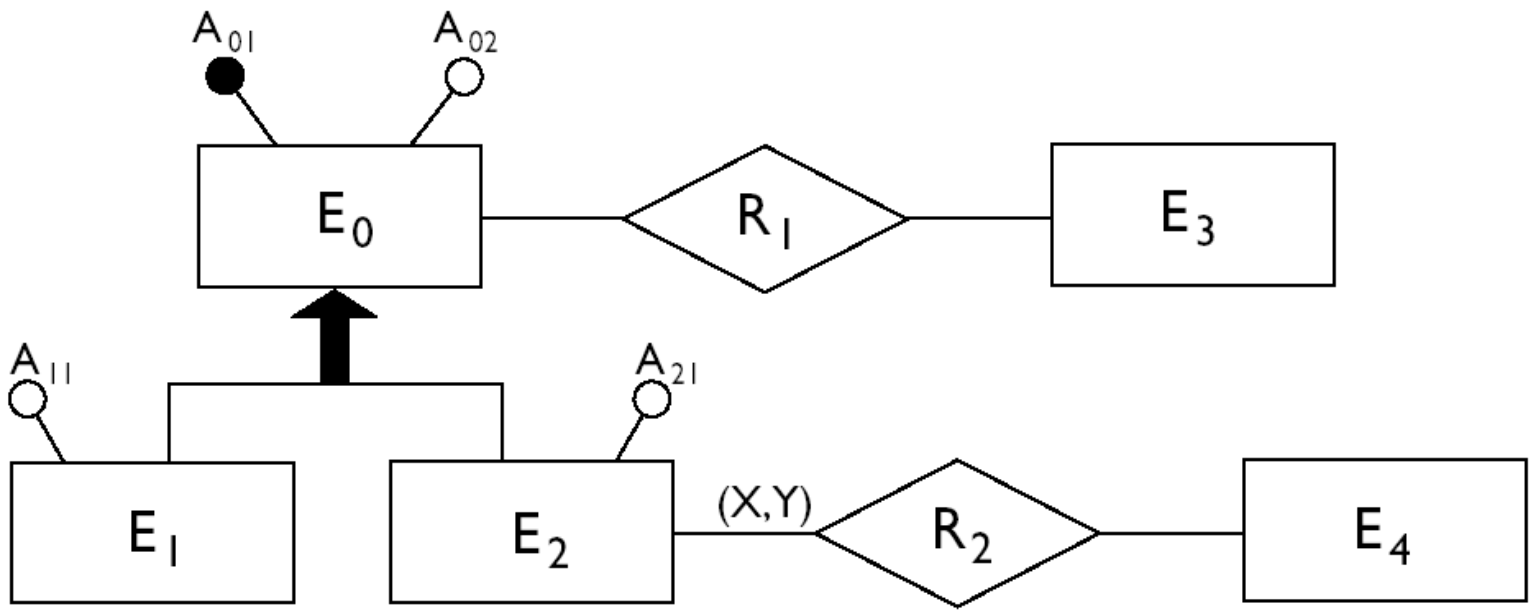
Esempio pratico

- In caso di assenza del dato ridondante, l'operazione 1 richiede:
  - 1 accesso in scrittura a PERSONA,
  - 1 accesso in scrittura a RESIDENZA.
- L'operazione 2 richiede invece:
  - 1 accesso in lettura a CITTA' (trascurabile),
  - 5000 accessi in lettura in media (num. Abitanti/num. città) a RESIDENZA.
- Quindi in questo caso (contando sempre doppio il costo delle operazioni in scrittura), abbiamo un costo totale di 12.000 accessi al giorno, risparmiando meno di 1 KB di memoria.
- Ciò deriva dal fatto che gli accessi in lettura necessari per calcolare il dato derivato sono molto maggiori di quelli in scrittura per mantenerlo aggiornato.

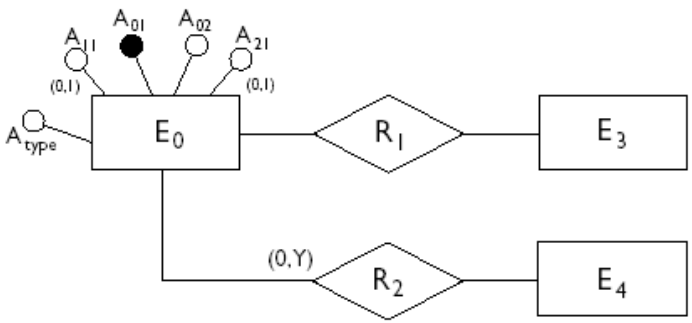
Eliminazione delle generalizzazioni

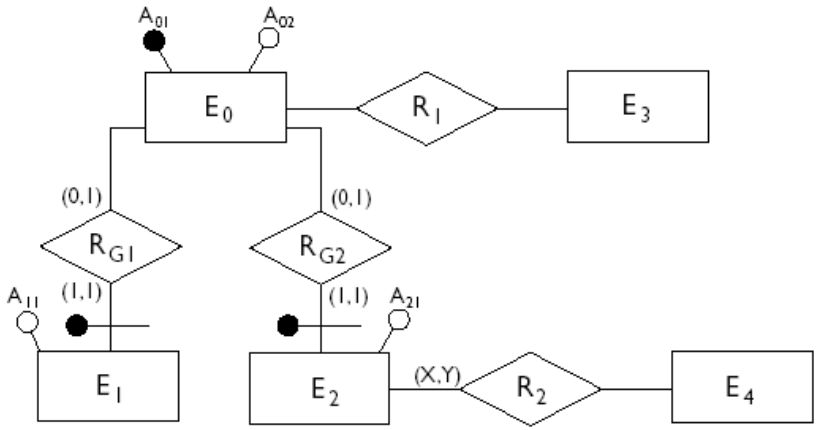
- Ci sono tre alternative possibili:
  - **Accorpamento delle entità figlie nell'entità padre**: le entità figlie spariscono ed i loro attributi e partecipazioni ad associazioni vengono aggiunti al padre; viene inoltre aggiunto a quest'ultimo un attributo che serve a distinguere il tipo di occorrenza (a seconda della "provenienza" dalle entità figlie).
  - **Accorpamento dell'entità padre nelle entità figlie**: l'entità padre sparisce ed i suoi attributi e le associazioni a cui partecipava vengono aggiunti alle entità figlie (ereditarietà).
  - **Sostituzione della generalizzazione con associazioni (uno-a-uno)**: non ci sono trasferimenti di attributi o associazioni, in quanto le figlie sono sempre identificate esternamente dall'entità padre, ma si rende necessario introdurre un vincolo che impedisca alle occorrenza del padre di partecipare a più di una delle nuove associazioni introdotte.

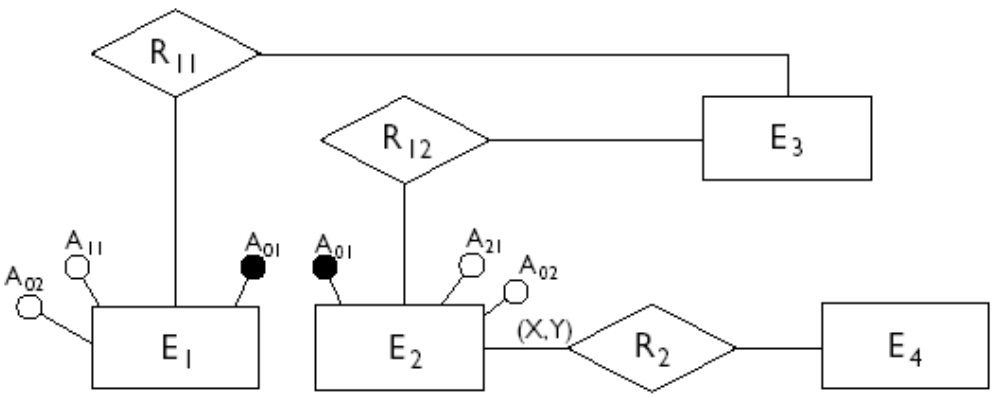
Esempio



Ristrutturazioni possibili







Eliminazione delle generalizzazioni

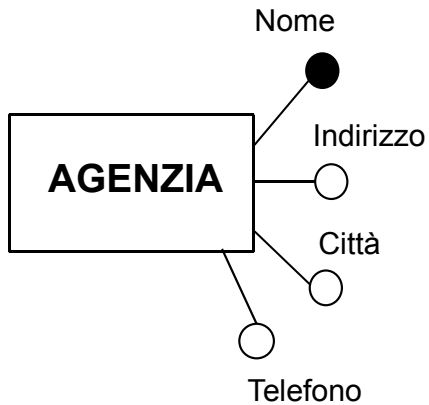
- Vantaggi/svantaggi delle alternative di eliminazione:
  - La prima alternativa è conveniente quando le operazioni non distinguono tra le occorrenze e gli attributi delle entità in gioco. C'è un aumento dello spreco di memoria (presenza di valori nulli) bilanciato da una diminuzione degli accessi necessari.
  - La seconda alternativa è possibile soltanto quando la generalizzazione è totale. Conviene quando le operazioni distinguono fra le entità figlie, usando soltanto occorrenze di una e non delle altre. In questo caso abbiamo sia un risparmio di memoria rispetto alla prima alternativa (assenza di valori nulli), sia una diminuzione degli accessi necessari (non è necessario visitare l'entità padre).
  - La terza alternativa conviene quando la generalizzazione non è totale e ci sono operazioni che distinguono fra entità padre ed entità figlie. Rispetto alla prima alternativa si risparmia memoria (per l'assenza di valori nulli), ma c'è un aumento degli accessi.
  - La terza alternativa ha il vantaggio (che esula dallo schema di analisi dei costi considerato) consistente nel fatto che genera entità con pochi attributi. Quindi le strutture logiche risultanti saranno di piccole dimensioni, consentendo di estrarre un elevato numero di risultati (tuple) ad ogni query.

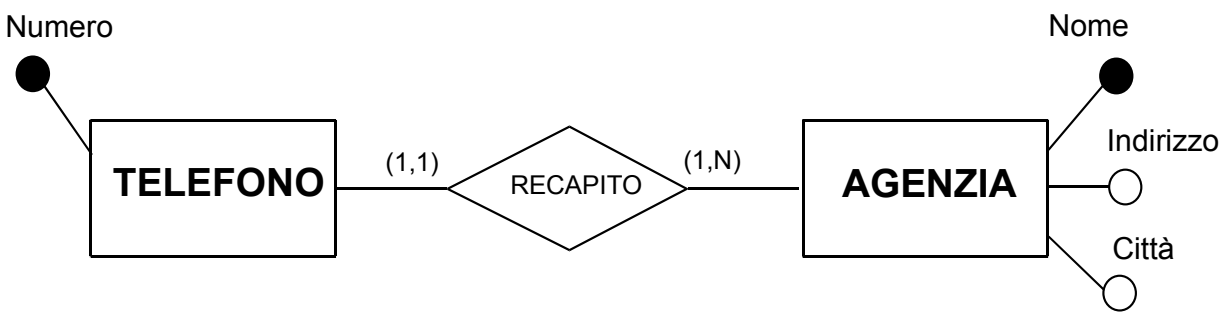
Partizionamento/accorpamento

- Per garantire una maggiore efficienza delle operazioni, entità e associazioni possono essere partizionate o accorpate in base al seguente principio:
  - Si riducono gli accessi separando gli attributi di uno stesso concetto che vengono utilizzati da operazioni distinte e raggruppando quelli di concetti diversi che vengono utilizzati dalle stesse operazioni.
- Per quanto riguarda i partizionamenti di entità si ha quanto segue:
  - I partizionamenti effettuati operando sugli attributi vengono detti *decomposizioni verticali*: hanno l'effetto di produrre entità con pochi attributi.
  - I partizionamenti effettuati operando sulle occorrenze di un'entità vengono detti *decomposizioni orizzontali*: possono avere delle ripercussioni negative sulle prestazioni del sistema in quanto comportano la potenziale duplicazione di tutte le relazioni a cui partecipava l'entità originale.

Eliminazione degli attributi multivalore

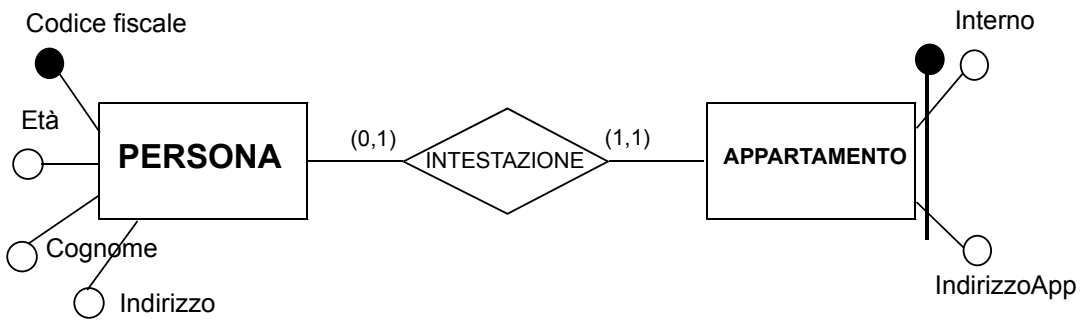
- Gli attributi multivalore vanno eliminati in quanto non hanno un corrispondente costrutto nel modello relazionale.

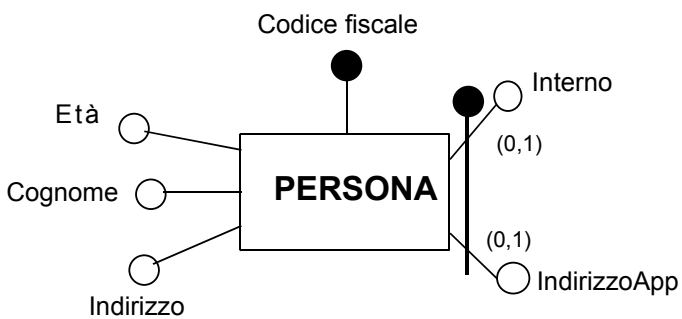




Accorpamento di entità

- Gli accorpamenti in genere si eseguono su entità che partecipano ad associazioni di tipo uno-a-uno. Negli altri casi (uno-a-molti e molti-a-molti) è facile che si generino ridondanze.
- Un possibile effetto collaterale di un accorpamento fra entità è la presenza di valori nulli.





Partizionamento/accorpamento di associazioni

- Quanto detto a proposito del partizionamento/accorpamento di entità può essere esteso anche alle associazioni.
- Il criterio da seguire è quello di decomporre un'associazione fra due entità in due o più associazioni fra le stesse entità per separare le occorrenze dell'associazione originale che erano sempre utilizzate in modo esclusivo dalle operazioni.
- Viceversa conviene accorpare due o più associazioni fra entità che si riferiscono ad aspetti diversi dello stesso concetto in un'unica associazione, se le varie occorrenze delle singole associazioni vengono sempre utilizzate contemporaneamente dalle operazioni.

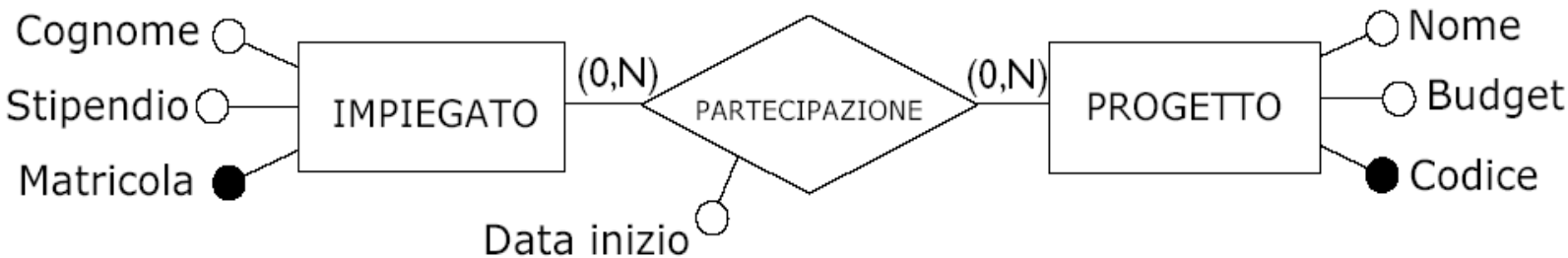
Identificatori principali

- Dato che le chiavi nel modello relazionale vengono usate per stabilire delle corrispondenze fra dati appartenenti a relazioni distinte, è fondamentale stabilire quali siano gli identificatori principali.
- Infatti questi ultimi corrisponderanno poi alle chiavi primarie sulle quali i DBMS costruiscono delle strutture ausiliarie (indici) per il reperimento efficiente dei dati.
- Quindi, nel caso vi siano entità con più identificatori, bisogna sceglierne uno principale in base ai seguenti criteri:
  - Vanno scartati gli attributi che ammettono valori nulli.
  - In genere è preferibile scegliere identificatori composti da pochi attributi.
  - Gli identificatori interni con pochi attributi vanno preferiti a quelli esterni.
  - Gli identificatori utilizzati da molte operazioni sono da preferire agli altri.
- Nel caso non vi siano degli identificatori idonei, è possibile introdurne uno apposito che conterrà come valori dei codici, generati appositamente per distinguere le varie occorrenze dell'entità.

Traduzione verso il modello relazionale

- Dato lo schema E-R ristrutturato si procede alla sua traduzione in uno schema logico *equivalente*, i.e., uno schema che rappresenta le stesse informazioni.
- I casi contemplati sono i seguenti:
  - Entità.
  - Associazioni molti-a-molti.
  - Associazioni uno-a-molti.
  - Entità con identificatore esterno.
  - Associazioni uno-a-uno.

Associazioni molti-a-molti

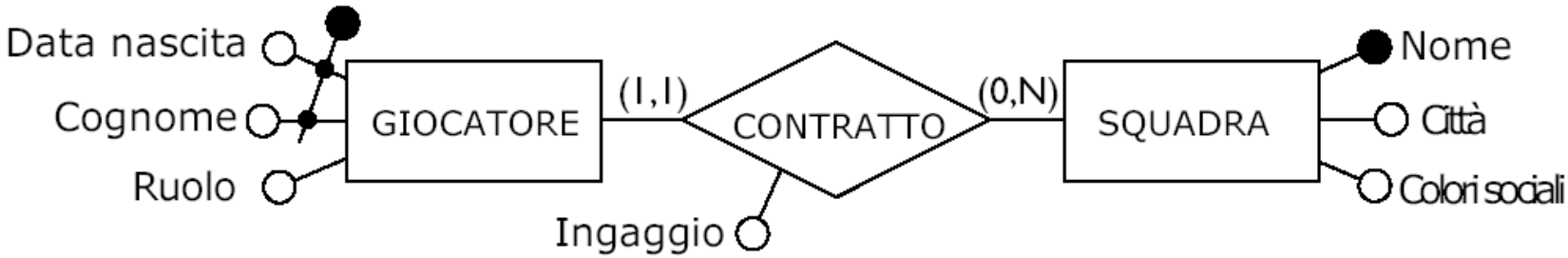


IMPIEGATO(Matricola,Cognome, Stipendio)
PROGETTO(Codice,Nome,Budget)
PARTECIPAZIONE(Matricola,Codice,DataInizio)

Al fine di rendere più comprensibile lo schema è opportuno eseguire delle rinomine:

PARTECIPAZIONE(Impiegato,Progetto,DataInizio)

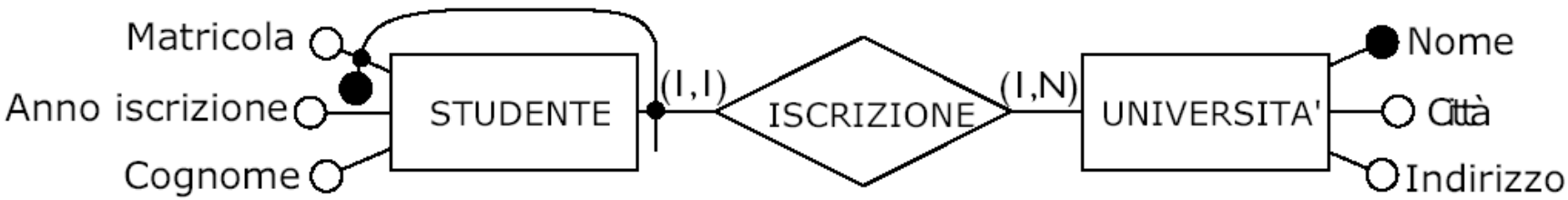
Associazioni uno a molti



In base alla modalità di traduzione per le associazioni molti-a-molti lo schema logico risultante dovrebbe essere il seguente:
GIOCATORE(Cognome,DataNascita, Ruolo)
SQUADRA(Nome,Città,ColoriSociali)
CONTRATTO(Giocatore,DataNascitaGiocatore,NomeSquadra,Ingaggio)

Tuttavia, siccome GIOCATORE E CONTRATTO hanno la stessa chiave è possibile fonderle, ottenendo lo schema seguente:
GIOCATORE(Cognome,DataNascita, Ruolo, NomeSquadra, Ingaggio)
SQUADRA(Nome,Città,ColoriSociali)

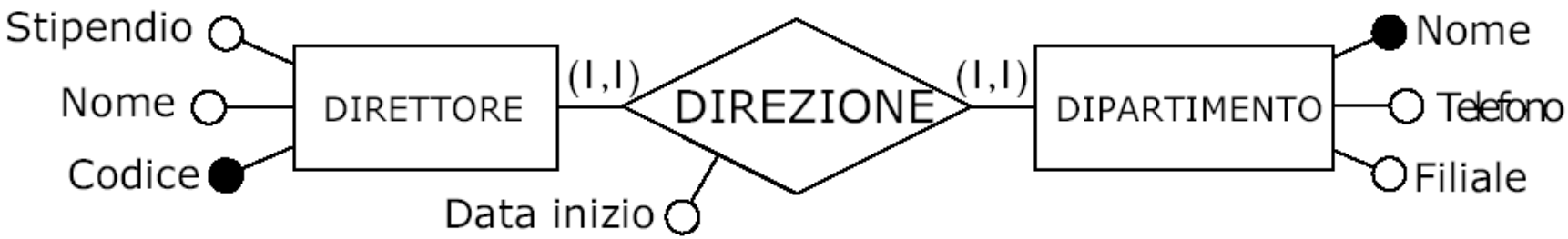
Entità con identificatore esterno



In questo caso, rappresentando l'identificatore esterno, si rappresenta automaticamente anche l'associazione fra le due entità:

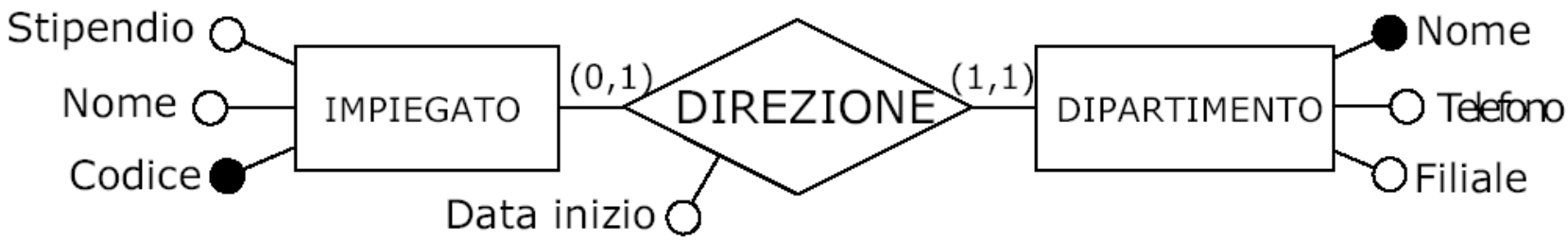
STUDENTE(Matricola, NomeUniversità, Cognome, AnnoIscrizione)
UNIVERSITA'(Nome, Città, Indirizzo)

Associazioni uno-a-uno



DIRETTORE(Codice,Nome,Stipendio)

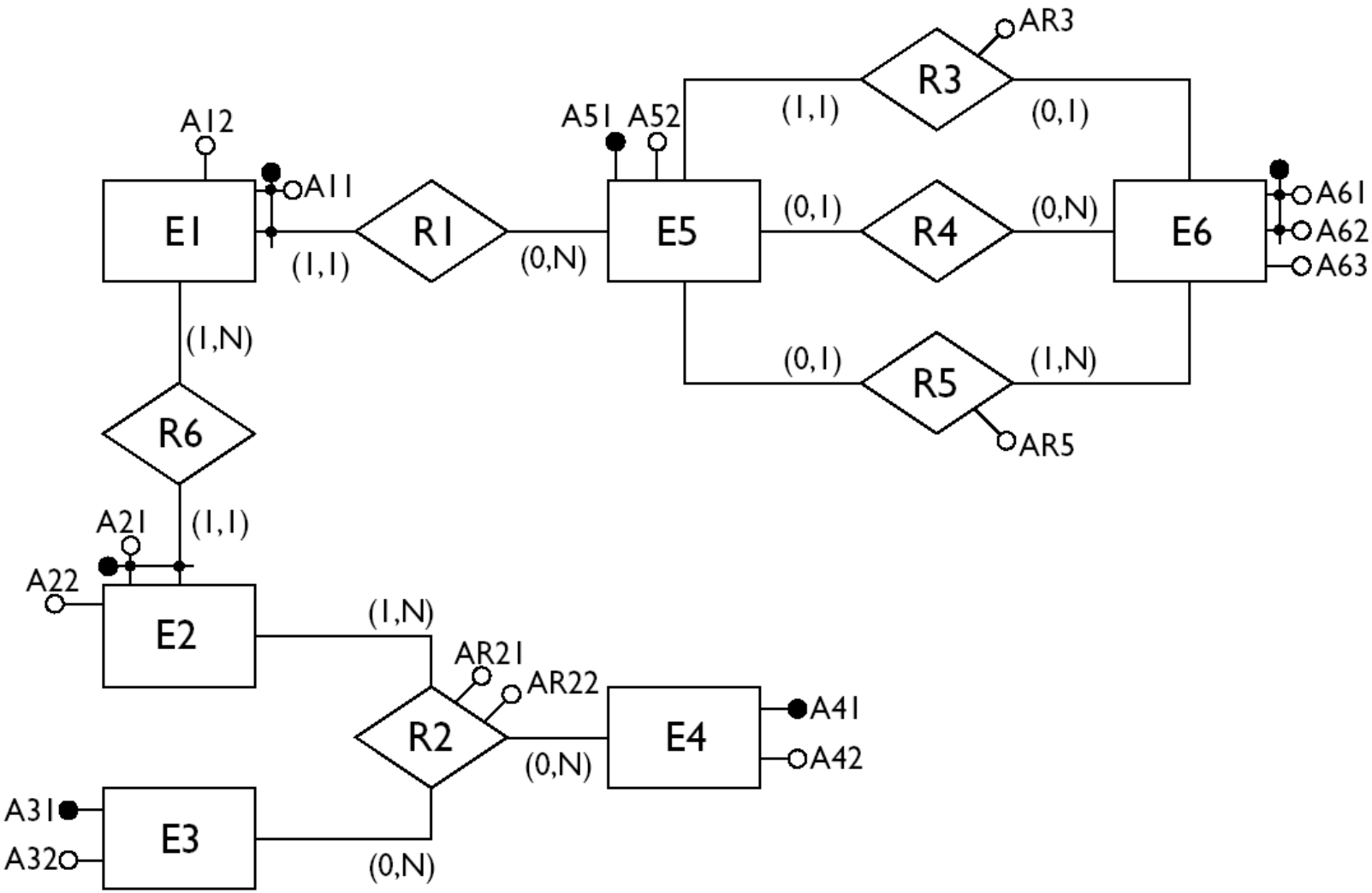
DIPARTIMENTO(Nome,Telefono,Filiale,Direttore,DataInizio)



IMPIEGATO(Codice,Nome,Stipendio)

DIPARTIMENTO(Nome,Telefono,Filiale,Direttore,DataInizio)

Esempio


AR3
R3
(1,1)
(0,1)
A51
A52
A12
A11
E1
R1
(1,1)
(0,N)
E5
(0,1)
R4
(0,N)
E6
A61
A62
A63
(1,N)
(0,1)
R5
(1,N)
AR5
R6
A21
(1,1)
A22
E2
(1,N)
AR21
AR22
R2
E4
A41
A42
(0,N)
A31
A32
E3
(0,N)

Esempio

Lo schema logico corrispondente allo schema E-R precedente è quindi definito come segue:

E1(A11, A51, A12)

E2(A21, A11, A51, A22)

E3(A31, A32)

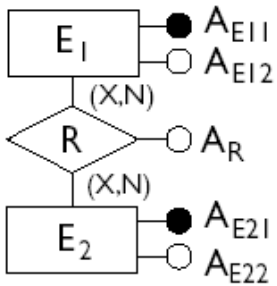
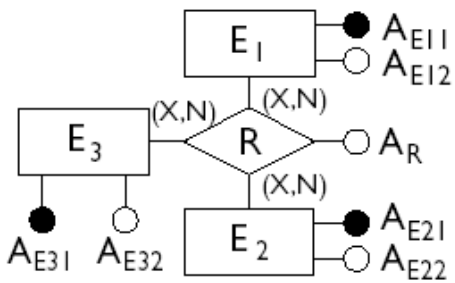
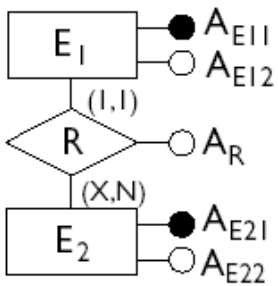
E4(A41,A42)

E5(A51, A52, A61R3, A62R3, AR3, A61R4, A62R4, A61R5, A62R5, AR5)

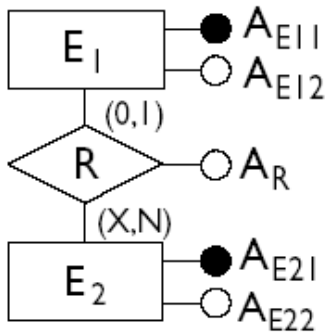
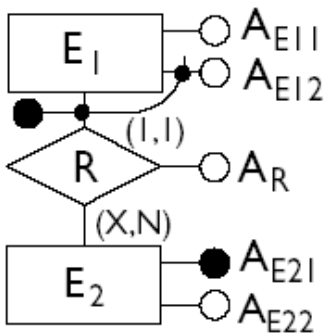
E6(A61, A62, A63)

R2(A21, A11, A51, A31, A41, AR21, AR22)

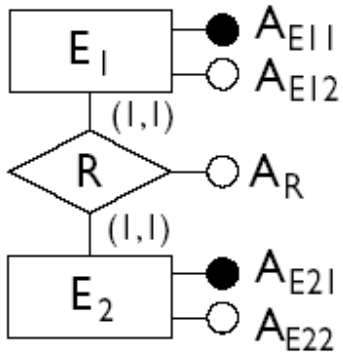
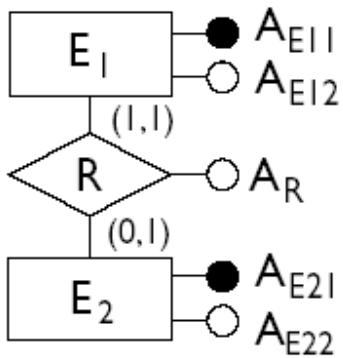
Schemi riassuntivi

Tipo	Concetto iniziale	Risultato possibile
Associazione binaria molti a molti		E1(AE11,AE12) E2(AE21,AE22) R(AE11, AE21,AR)
Associazione ternaria molti a molti		E1(AE11,AE12) E2(AE21,AE22) E3(AE31,AE32) R(AE11, AE21, AE31,AR)
Associazione uno a molti con partecipazione obbligatoria		E1(AE11,AE12,AE21,AR) E2(AE21,AE22)

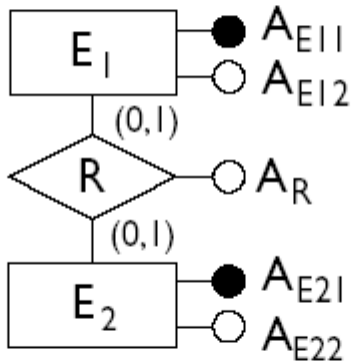
Schemi riassuntivi

Tipo	Concetto iniziale	Risultato possibile
Associazione uno a molti con partecipazione opzionale		$E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}}, \underline{A_{E21}}, A_R)$ Oppure: $E_1(\underline{A_{E11}}, A_{E21}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$
Relazione con identificatore esterno		$E_1(\underline{A_{E12}}, \underline{A_{E21}}, A_{E11}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

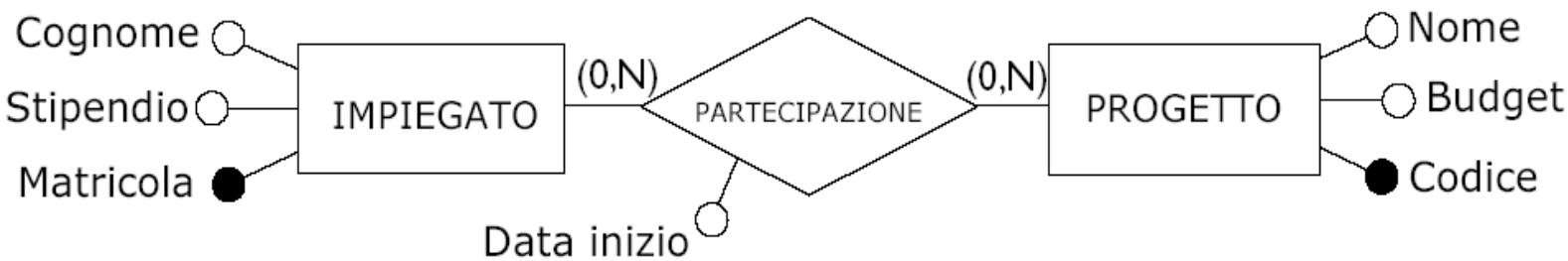
Schemi riassuntivi

Tipo	Concetto iniziale	Possibile risultato
Associazione uno a uno con partecipazione obbligatoria per entrambe le entità		$E_1(\underline{A_{E11}}, A_{E12}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Oppure: $E_2(\underline{A_{E21}}, A_{E22}, \underline{A_{E11}}, A_R)$ $E_1(\underline{A_{E11}}, A_{E12})$
Associazione uno a uno con partecipazione opzionale per entrambe le entità		$E_1(\underline{A_{E11}}, A_{E12}, \underline{A_{E21}}, A_R)$ $E_2(\underline{A_{E21}}, A_{E22})$

Schemi riassuntivi

Tipo	Concetto iniziale	Possibile risultato
Associazione uno a uno con partecipazione opzionale per entrambe le entità		$E_1(\underline{A_{E11}}, A_{E21})$ $E_2(\underline{A_{E21}}, A_{E22}, A^*_{E11}, A^*_R)$ Oppure: $E_1(\underline{A_{E11}}, A_{E12}, A^*_{E21}, A^*_R)$ $E_2(\underline{A_{E21}}, A_{E22})$ Oppure: $E_1(\underline{A_{E11}}, A_{E12})$ $E_2(\underline{A_{E21}}, A_{E22})$ $R(\underline{A_{E11}, A_{E21}}, A_R)$

Documentazione di schemi logici



IMPIEGATO(Matricola,Cognome, Stipendio)
PROGETTO(Codice,Nome,Budget)
PARTECIPAZIONE(Impiegato,Progetto,DataInizio)