

# La Progettazione di una basi di dati - Parte II

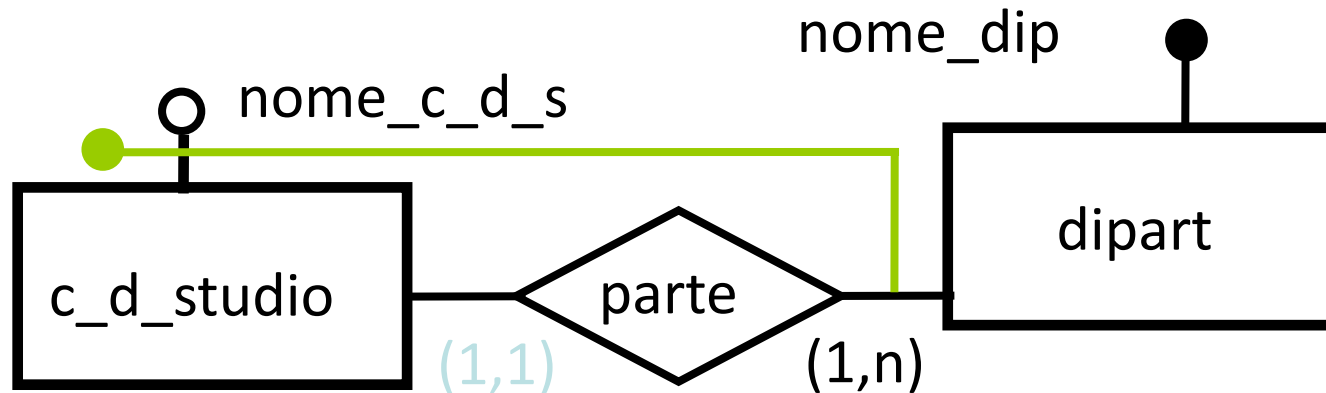
Prof. Alfredo Pulvirenti

Prof. Salvatore Alaimo

(Atzeni-Ceri Capitolo 7-8)

# Identificazione esterna

In alcuni casi una entità può essere identificata da altre ad essa collegate



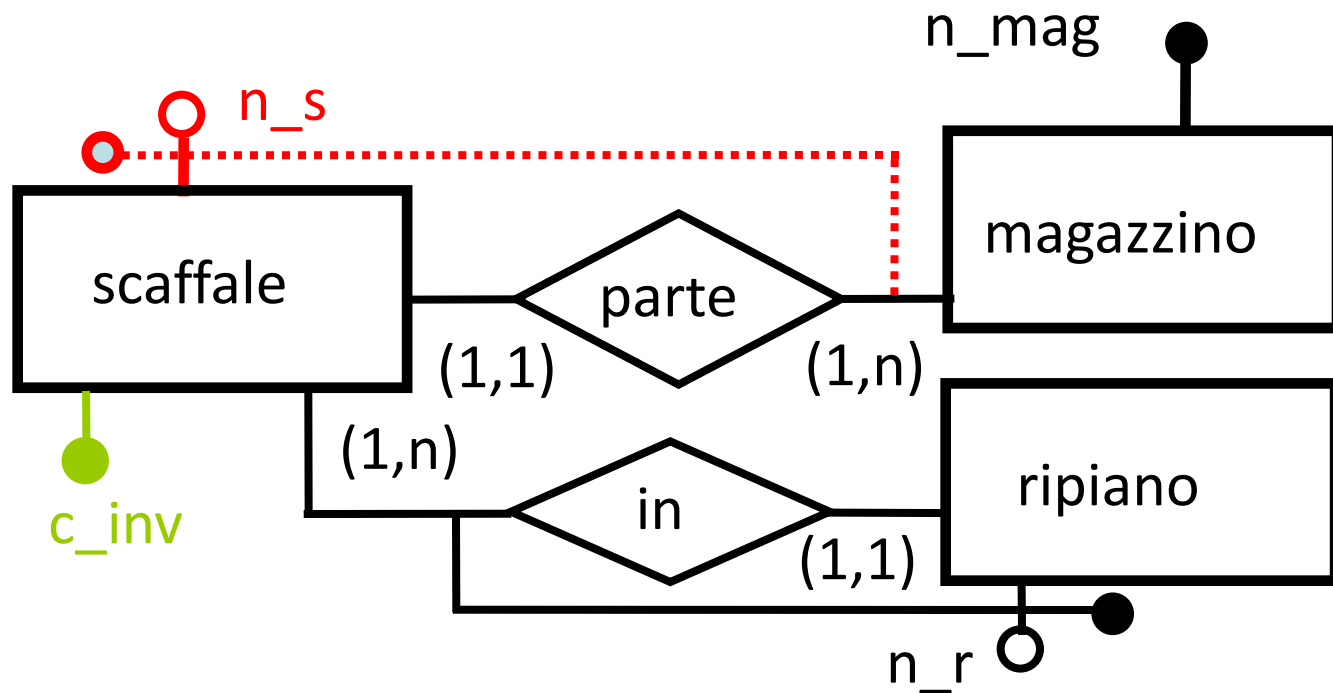
Nell'esempio i corsi di studio sono identificati da un nome proprio e da quello del dipartimento che li eroga, ad esempio:  
laurea in Informatica del dipartimento di Matematica e Informatica

# Regole da rispettare

- le identificazioni esterne avvengono sempre tramite associazioni binarie in cui l'entità da identificare partecipa con cardinalità (1,1)
- una identificazione esterna può coinvolgere una entità che a sua volta è identificata esternamente a patto che non si creino cicli di identificazione

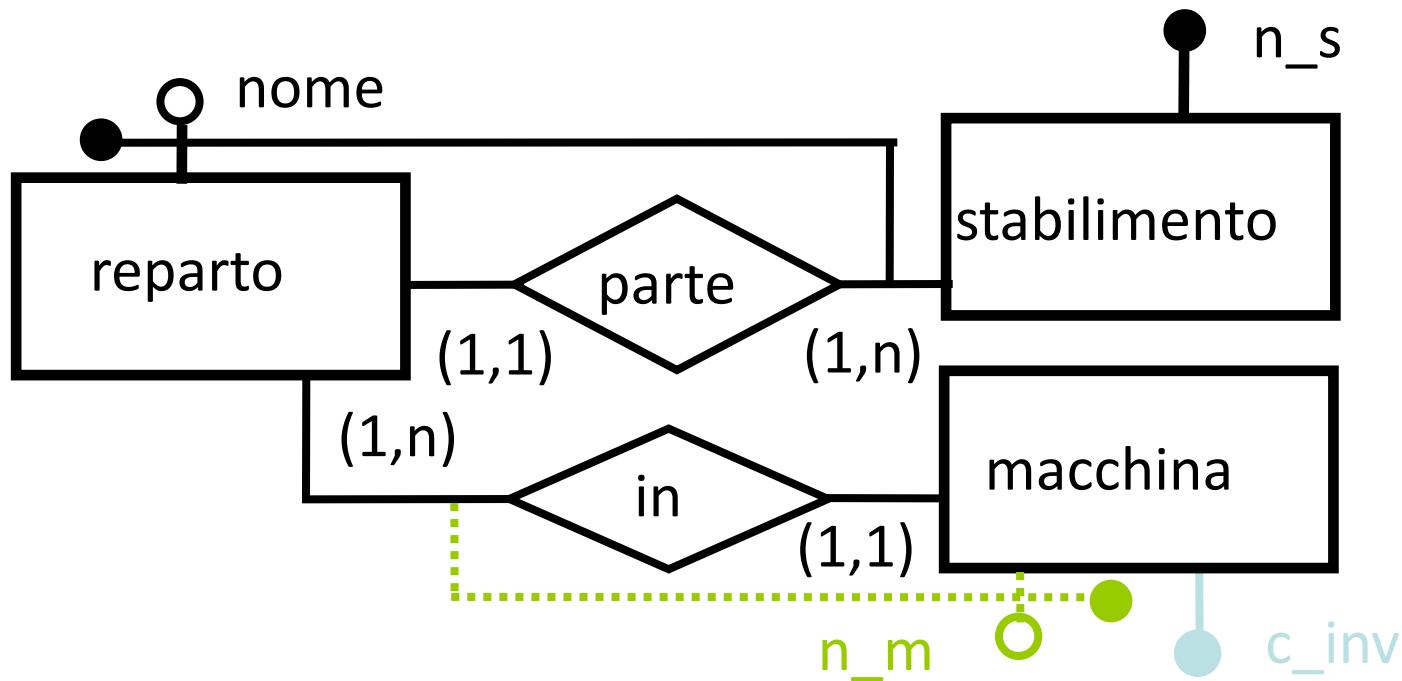
# Chiavi alternative

entità con chiavi alternative: interno ed esterna



# Scelta delle chiavi

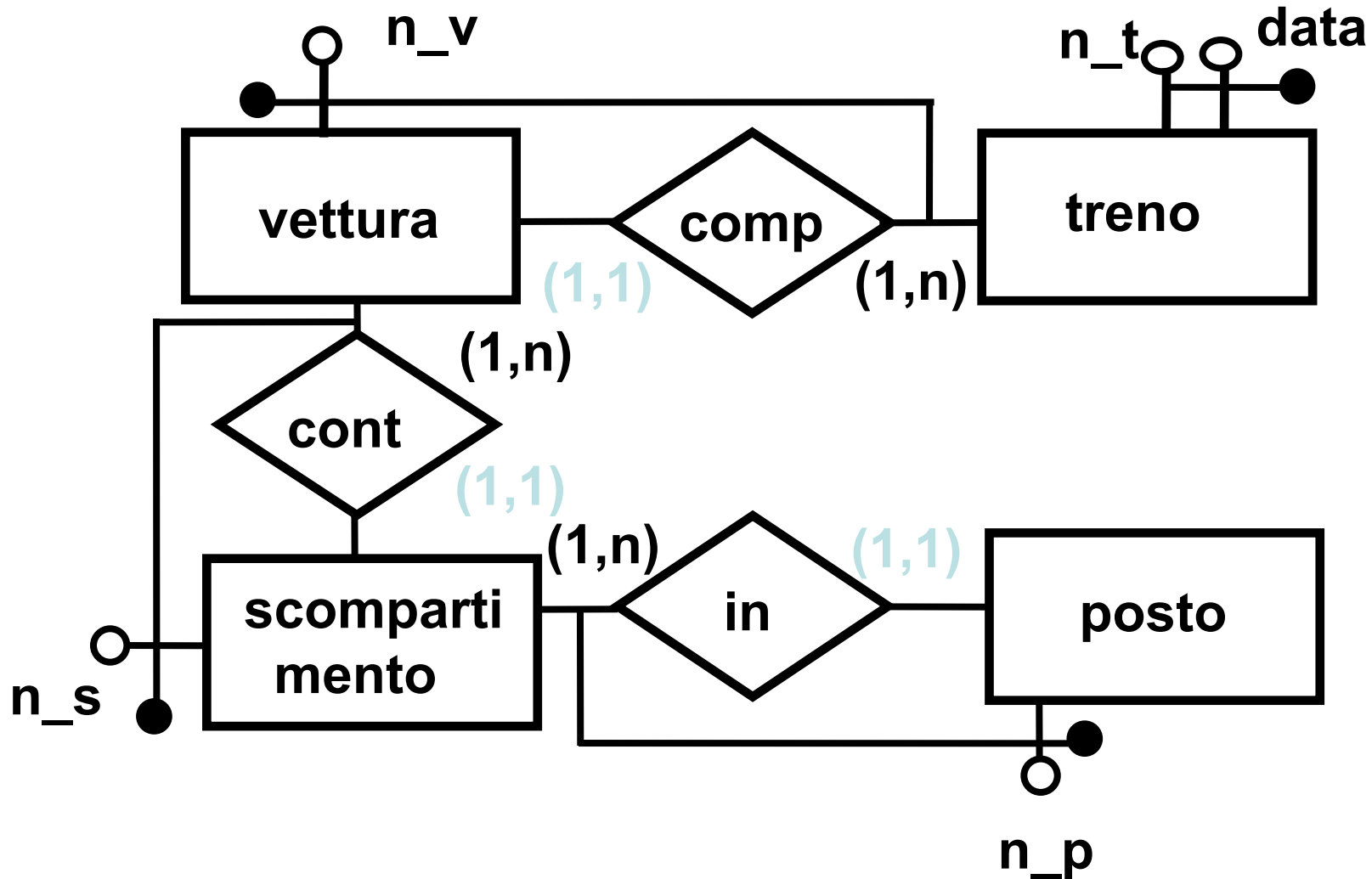
entità con chiavi alternative: interno ed esterna



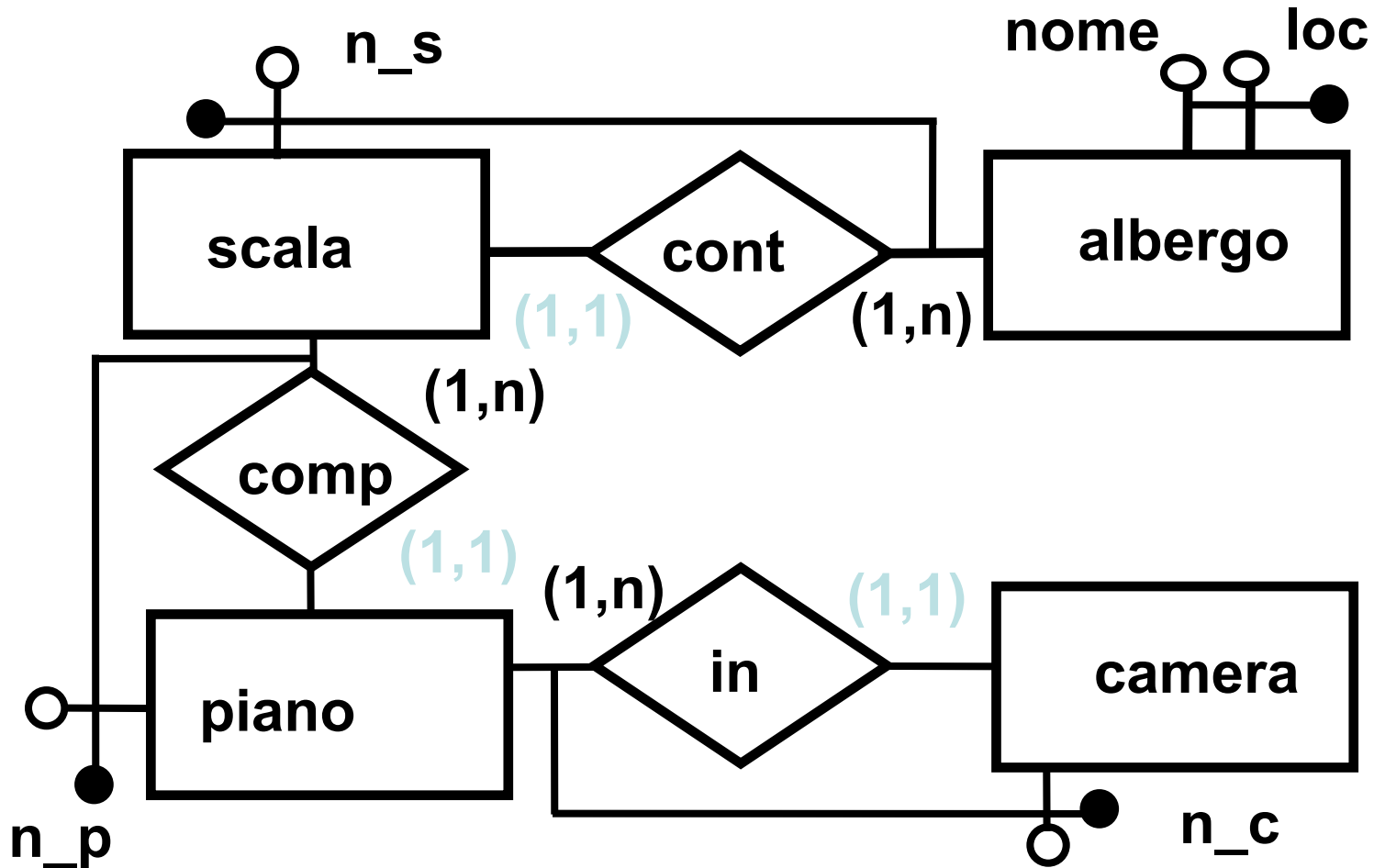
# Esempio: composizione treni

- i treni sono identificati da un codice e da una data, sono composti da vetture che contengono i posti da prenotare
- le vetture sono numerate, i posti sono numerati nello stesso modo all'interno di ogni vettura
- (potremmo tenere conto anche degli scompartimenti interni alle vetture)

# Schema composizione treni



# Esempio: camere d'albergo





# Gerarchie

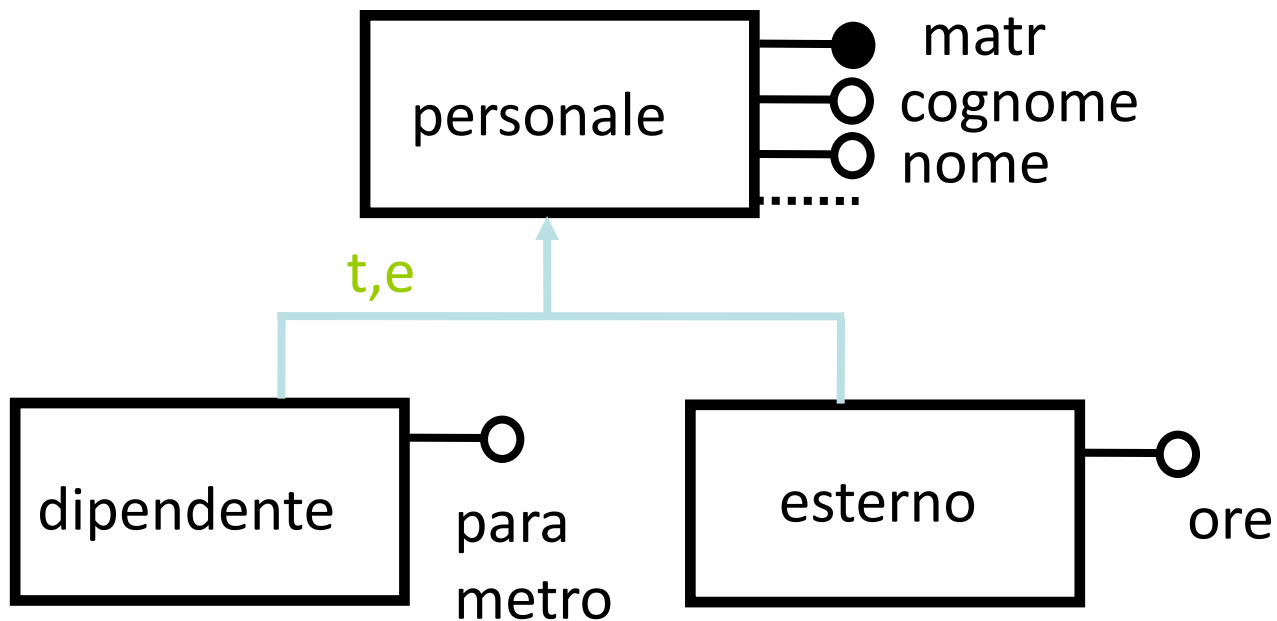
- spesso nella analisi di un settore aziendale può risultare che più entità risultino simili o casi particolari l'una dell'altra, derivanti da “viste” diverse da parte dell'utenza
- emerge quindi la necessità di evidenziare sottoclassi di alcune classi
- si definisce pertanto gerarchia di specializzazione il legame logico che esiste tra classi e sottoclassi

# Le gerarchie

- Definizione: la **gerarchia concettuale** è il legame logico tra un'entità padre  $E$  ed alcune entità figlie  $E_1 E_2 \dots E_n$  dove:
  - $E$  è la **generalizzazione** di  $E_1 E_2 \dots E_n$
  - $E_1 E_2 \dots E_n$  sono **specializzazioni** di  $E$
  - una istanza di  $E_k$  **è** anche istanza di  $E$  (e di tutte le sue generalizzazioni)
  - una istanza di  $E$  **può** essere una istanza di  $E_k$
  - NOTA: nel caso in cui  $n=1$  allora  $E_1$  è un **sottoinsieme** di  $E$

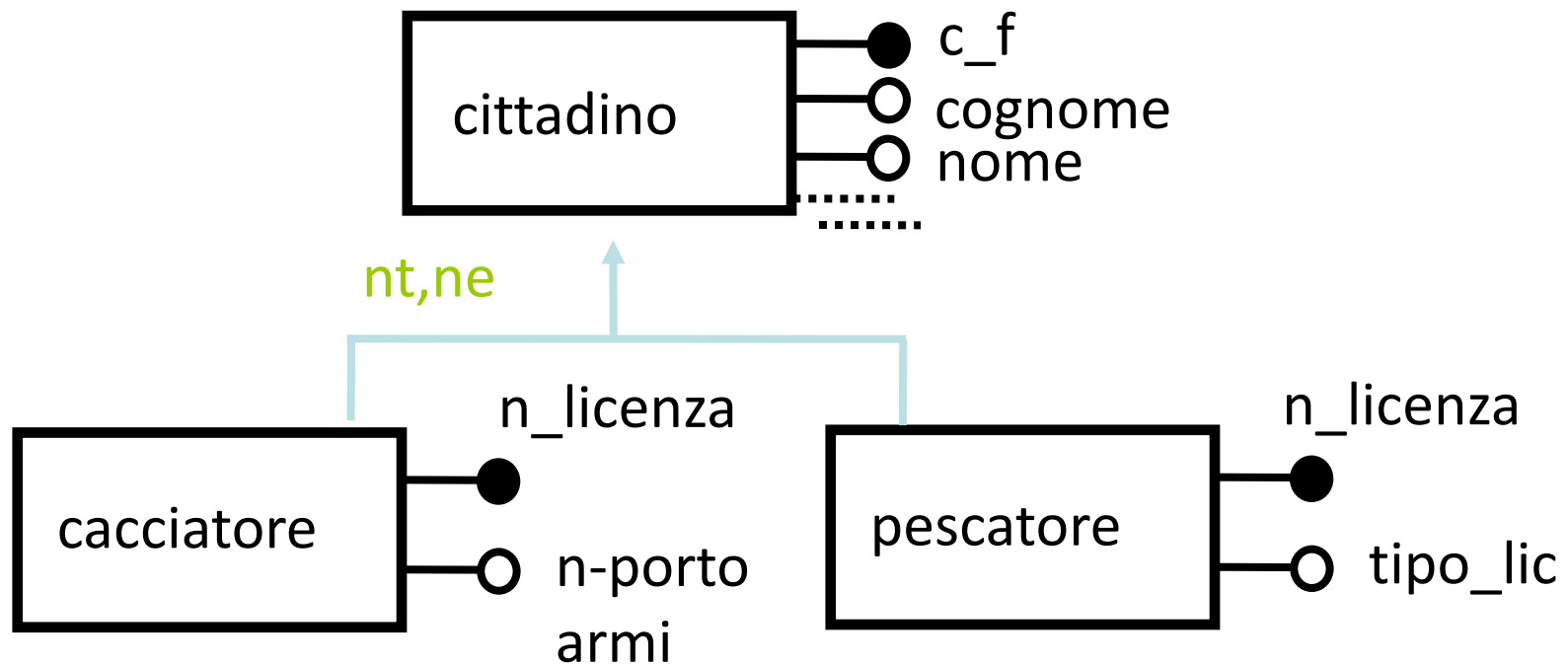
# Classificazione del personale

un'azienda si avvale dell'opera di professionisti esterni, quindi il suo personale si suddivide in esterni e dipendenti:



# Anagrafe comunale

un comune gestisce l'anagrafe ed i servizi per i suoi cittadini alcuni di questi richiedono i dati relativi alla licenza di pesca e/o di caccia:



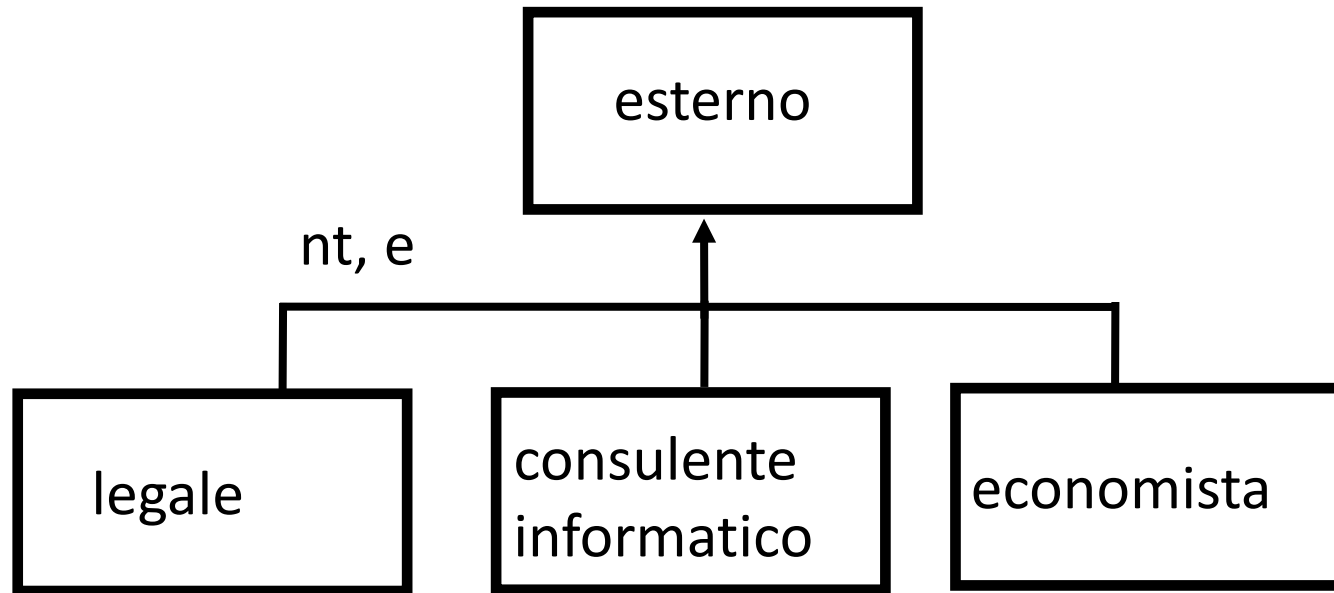
# Tipi di gerarchie: totalità

- **t** sta per **totale**: ogni istanza dell'entità padre deve far parte di una delle entità figlie
  - nell'esempio il personale si divide (completamente) in esterni e dipendenti
- **nt** sta per **non totale**: le istanze dell'entità padre possono far parte di una delle entità figlie
  - nell'esempio i pescatori sono un sottoinsieme dei cittadini

# Tipi di gerarchie: esclusività

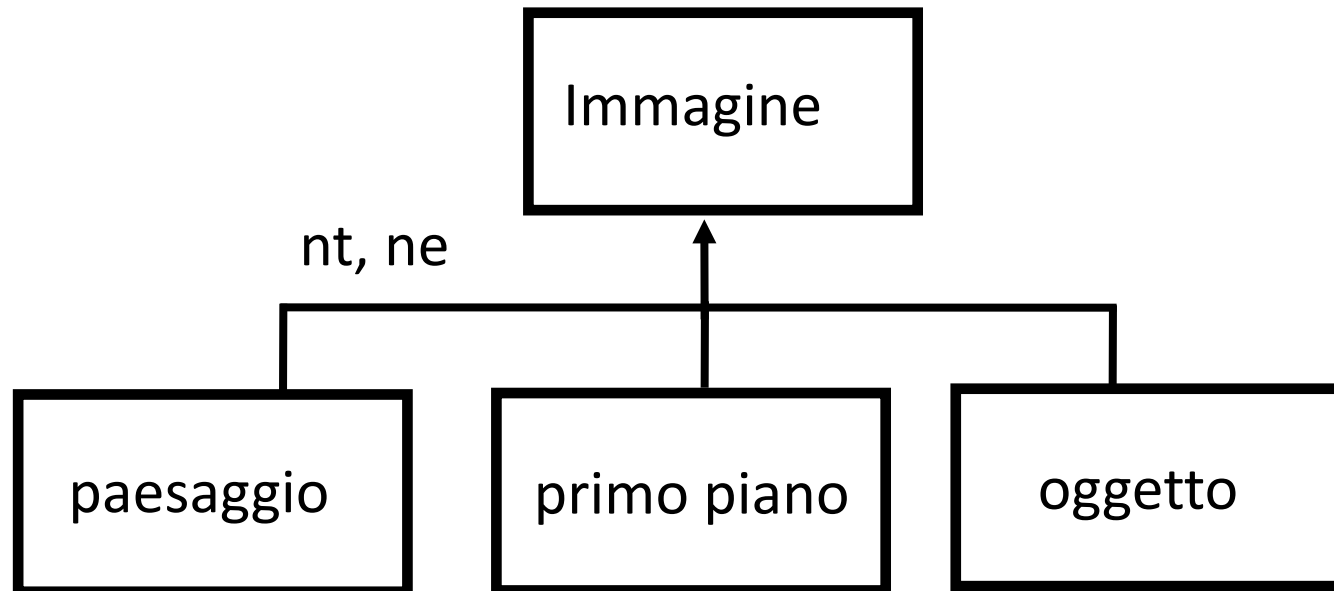
- **e** sta per **esclusiva**: ogni istanza dell'entità padre deve far parte di una sola delle entità figlie
  - esempio: una istanza di personale non può sia essere sia dipendente che esterno
- **ne** sta per **non esclusiva**: ogni istanza dell'entità padre può far parte di una o più entità figlie
  - esempio: un cittadino può essere sia pescatore che cacciatore

# Mansioni esterne



nt : **possono** esistere esterni generici che non sono né legali, né ingegneri, né economisti ma non interessa stabilire una **sottoclasse** ad hoc

# Tipi di immagini



*ne* : **possono** esistere immagini che al loro interno contengono primi piani, oggetti e paesaggio quindi le tre categorie **non si escludono**



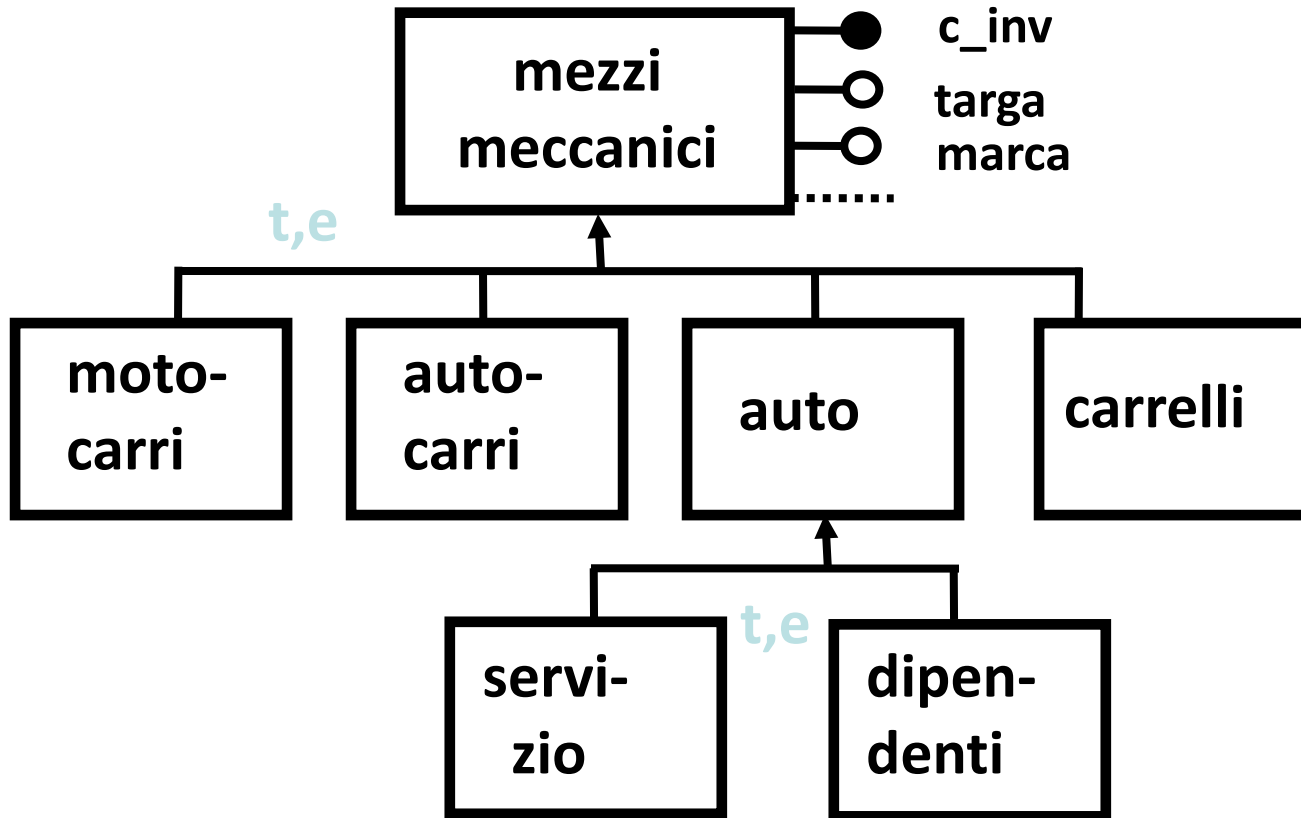
# Ereditarietà delle proprietà

- le *proprietà* dell'entità padre non devono essere replicate sull'entità figlia in quanto questa le *eredita* cioè:
- le proprietà dell'entità padre fanno parte del *tipo* dell'entità figlia
- non è vero il viceversa
  - il tipo di **personale** è: (**matricola**, **cognome**, **nome**, **indirizzo**, **data\_nascita**)

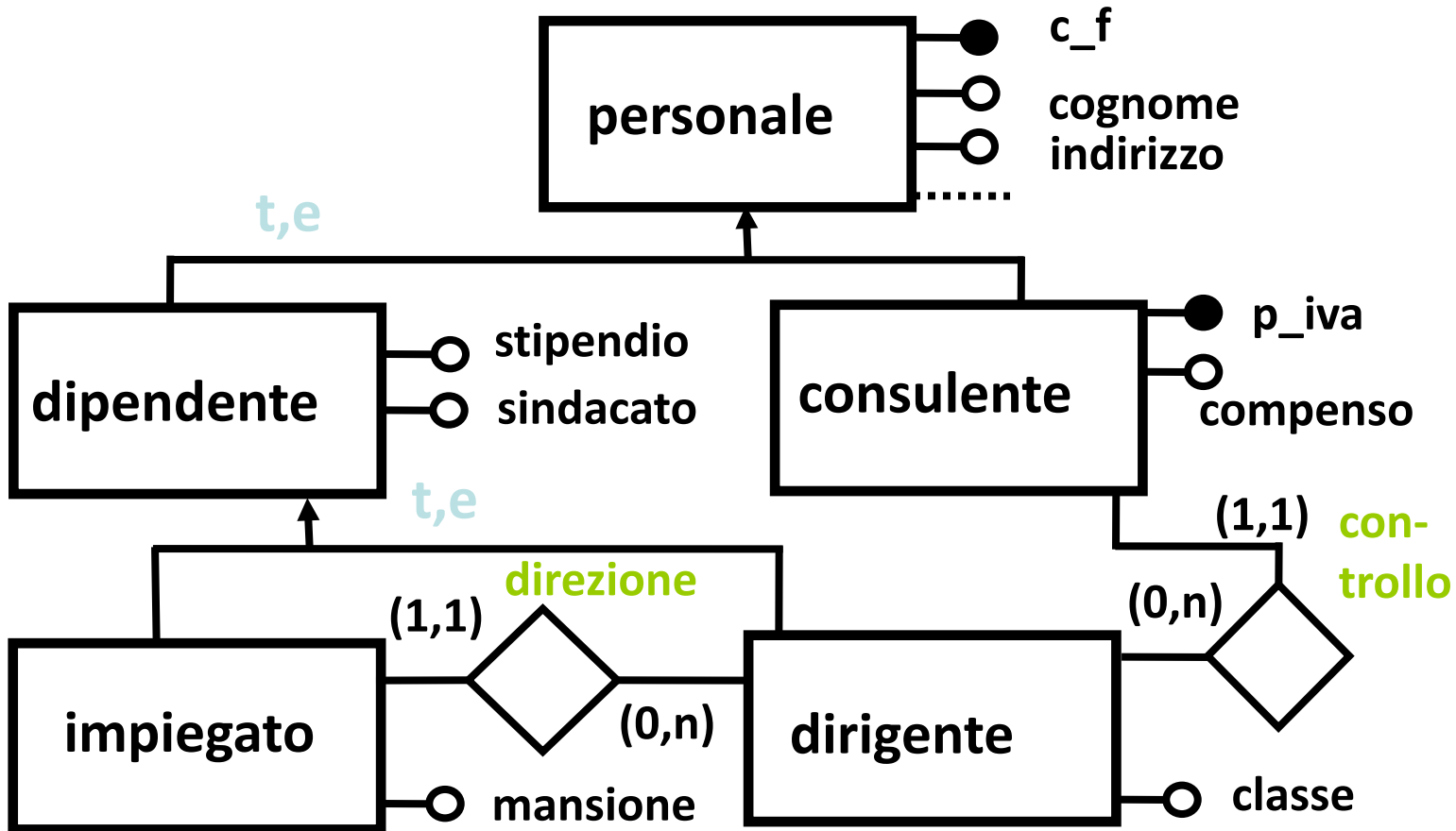
# Ereditarietà

- il tipo di dipendente è: (matricola, cognome, nome, indirizzo, data\_nascita, parametro)
- il tipo di esterno è: (matricola, cognome, nome, indirizzo, data\_nascita, ore)
- dipendente ed esterno hanno lo stesso tipo se considerati come personale
- NB: le gerarchie concettuali sono anche denominate gerarchie ISA
  - dipendente è un (is a ) personale
  - esterno è un (is a ) personale

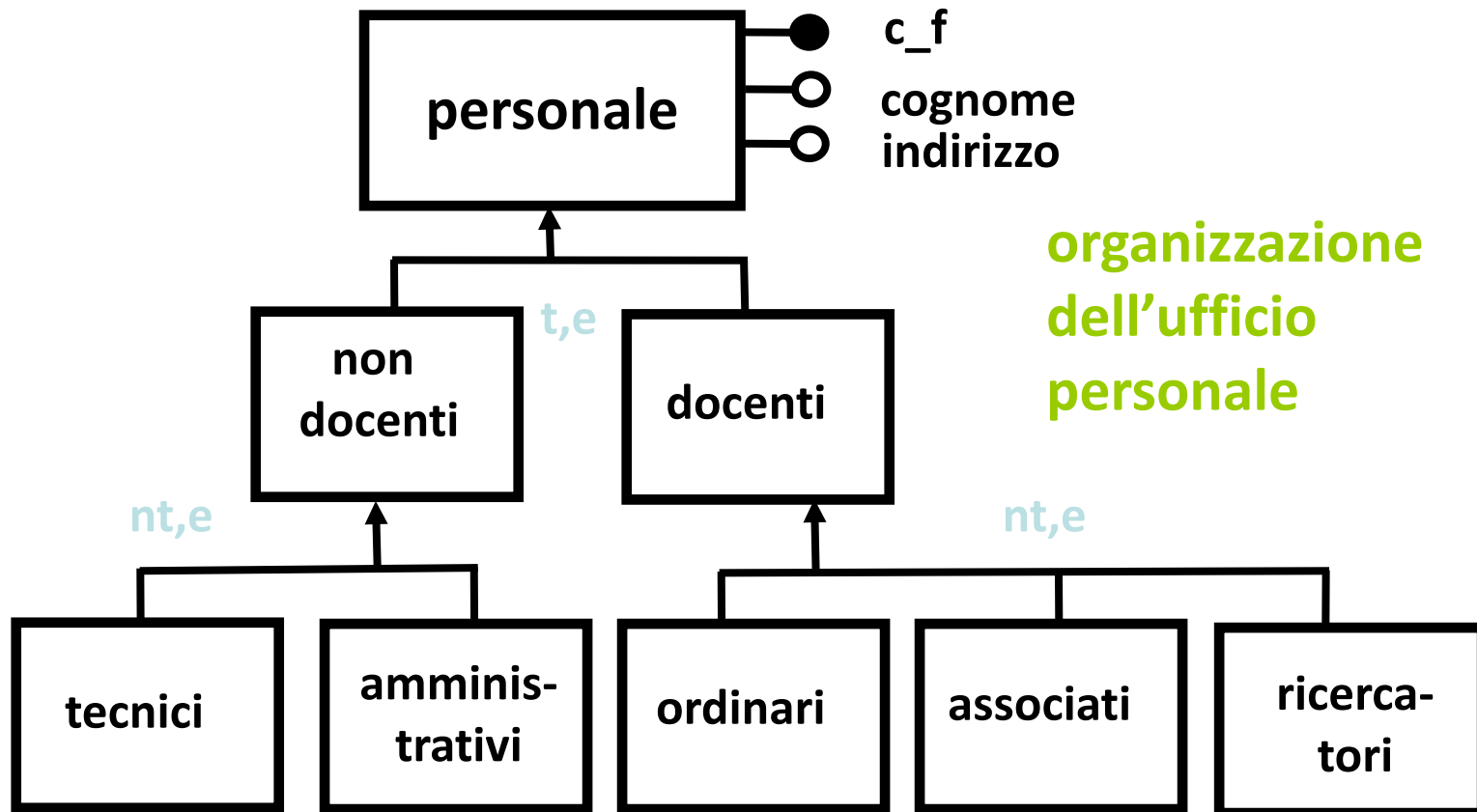
# Parco mezzi meccanici



# Anagrafe aziendale



# Università



# La Documentazione di schemi E-R

- Corredare lo schema E-R con una documentazione di supporto, per facilitare l'interpretazione dello schema e descrivere proprietà dei dati non espresse nello schema: es. (lo stipendio del dipendente non può essere maggiore di quello del direttore)
  - Definiamo le ***business rules***
    - *Descrizione di un concetto*
    - *Vincolo di integrità*
    - *Derivazione (un concetto che può essere ottenuto tramite calcoli su altri concetti)*

- *Descrizione di un concetto*
  - si esprime con il linguaggio naturale
- *Vincolo di integrita' (RV)*
  - *Concetto **deve/non deve** espressione sui concetti*  
(il direttore deve afferire a quel dipartimento)
  - *Derivazione (un concetto che puo' essere ottenuto tramite calcoli su altri concetti)*
    - *Concetto **si ottiene** operazione su concetti*  
(il numero degli impiegati di un dipartimento si ottiene contando gli impiegati che vi afferiscono)

# Documentazione da produrre

- 4 Tabelle
  - Il dizionario dei dati:
    - Una tabella per la specifica dei termini (entita', descrizione, attributi, identificatore)
    - Una tabella per la specifica delle relazioni (relazione, descrizione, entita' coinvolte e le rispettive cardinalita', attributi)
  - Regole di Vincolo
  - Regole di derivazione



# Progettazione Concettuale

- Cose da fare per avere una specifica dei requisiti piu' precisa e senza ambiguita'
- Scegliere il corretto livello di astrazione
  - Evitare termini troppo generici o troppo specifici
- Standardizzare la struttura delle frasi
  - Per dato rappresentiamo insieme di proprieta'
- Evitare frasi contorte
  - lavoratori dipendenti
- Individuare sinonimi/omonimi e unificare i termini
- Rendere esplicito il riferimento tra termini
  - Alcune proprieta' possono essere di alcuni dipendenti e non di tutti i tipi di dipendenti
- Costruire il glossario dei termini e l'elenco delle operazioni da effettuare
  - Tabella (termine, descrizione, sinonimi. collegamenti)
  - Lista operazione 1:.. , operazione : .. , etc..

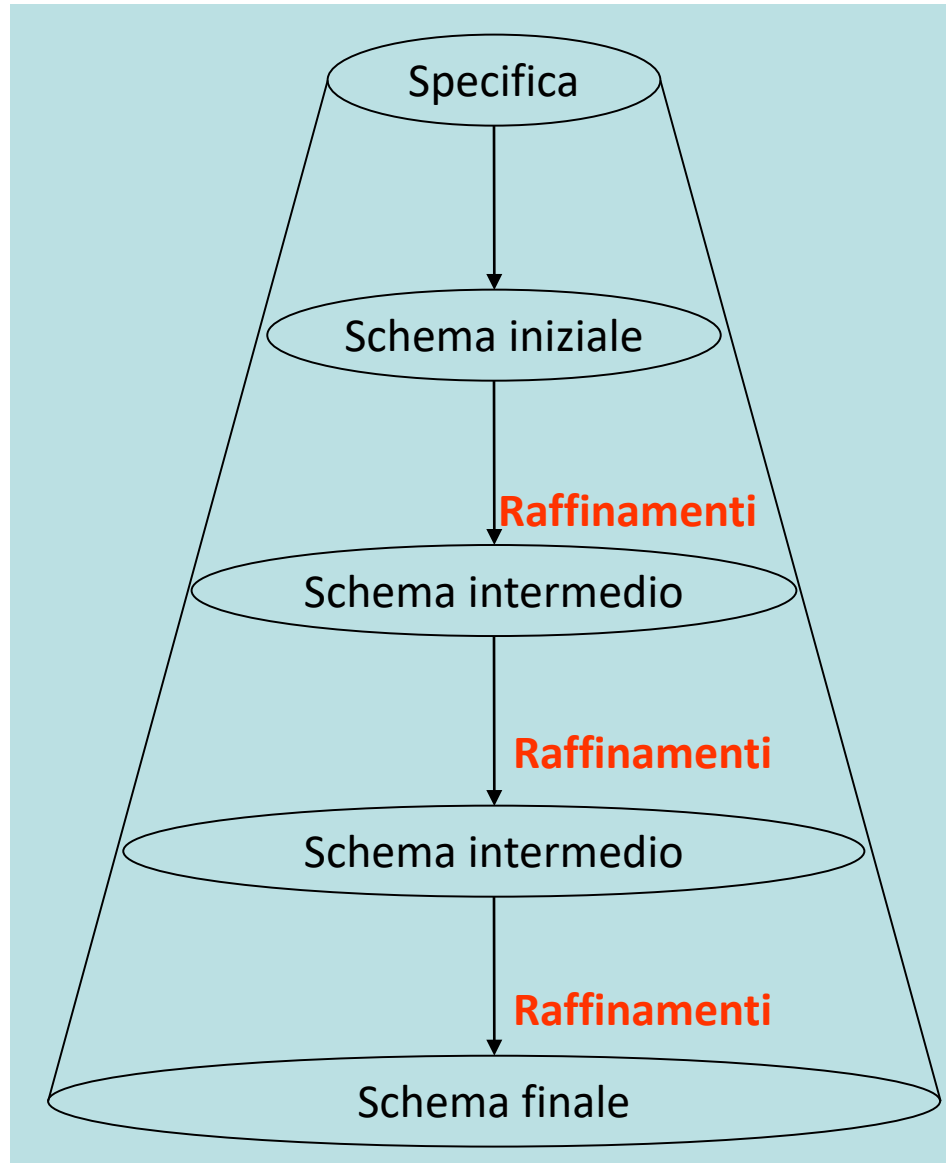
# Strategie di progetto

Top-Down

Bottom-Up

Generale

# Strategia Top-Down



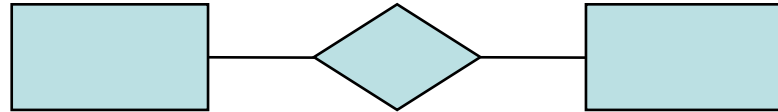
# Primitive di Trasformazione TOP-DOWN

- Le primitive di trasformazione top-down sono regole che operano su un singolo concetto dello schema e lo trasformano in una struttura più complessa che descrive il concetto con maggiore dettaglio

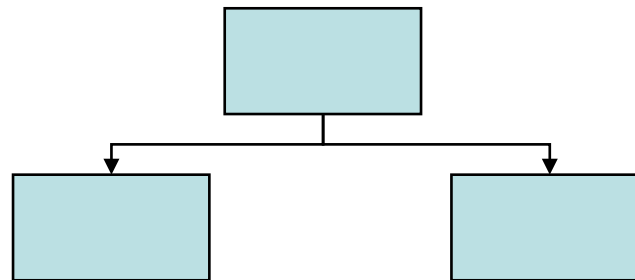
# Regole

- **Trasformazione**

- **T1:** si applica quando un'entità descrive *due concetti diversi* legati fra di loro.



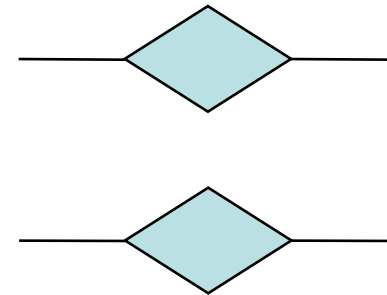
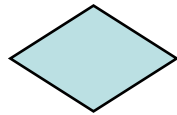
- **T2:** Un'entità è composta da **sotto-entità distinte**.



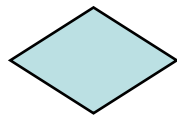
# Regole

- **Trasformazione**

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

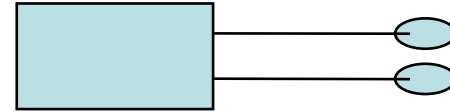


- **T4:** Una **relazione** descrive un concetto con **esistenza autonoma**. In questo caso essa va sostituita con un'entità.

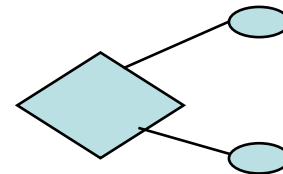
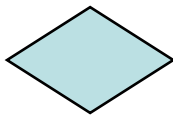


# Regole

- **Trasformazione**
  - **T5:** Si applica per aggiungere **attributi ad entità**.

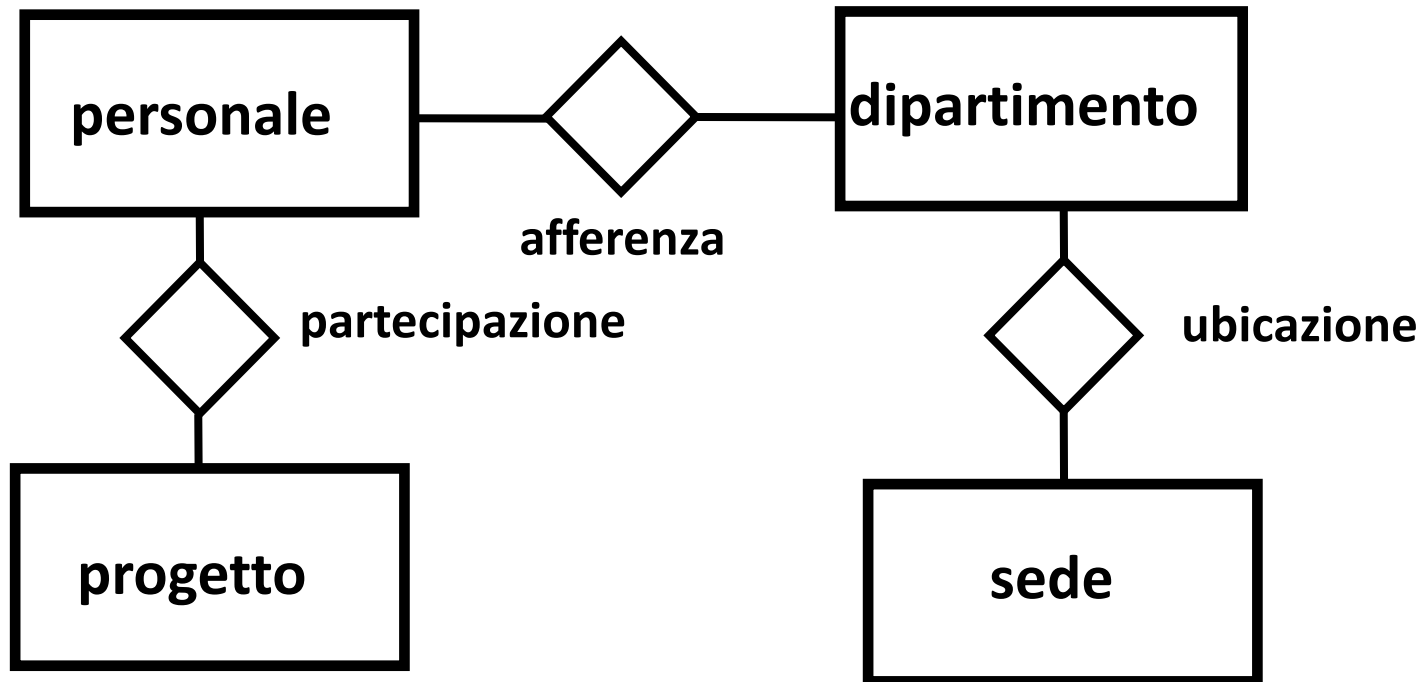


- **T6:** Si applica per aggiungere **attributi a relazioni**.



# Esempio

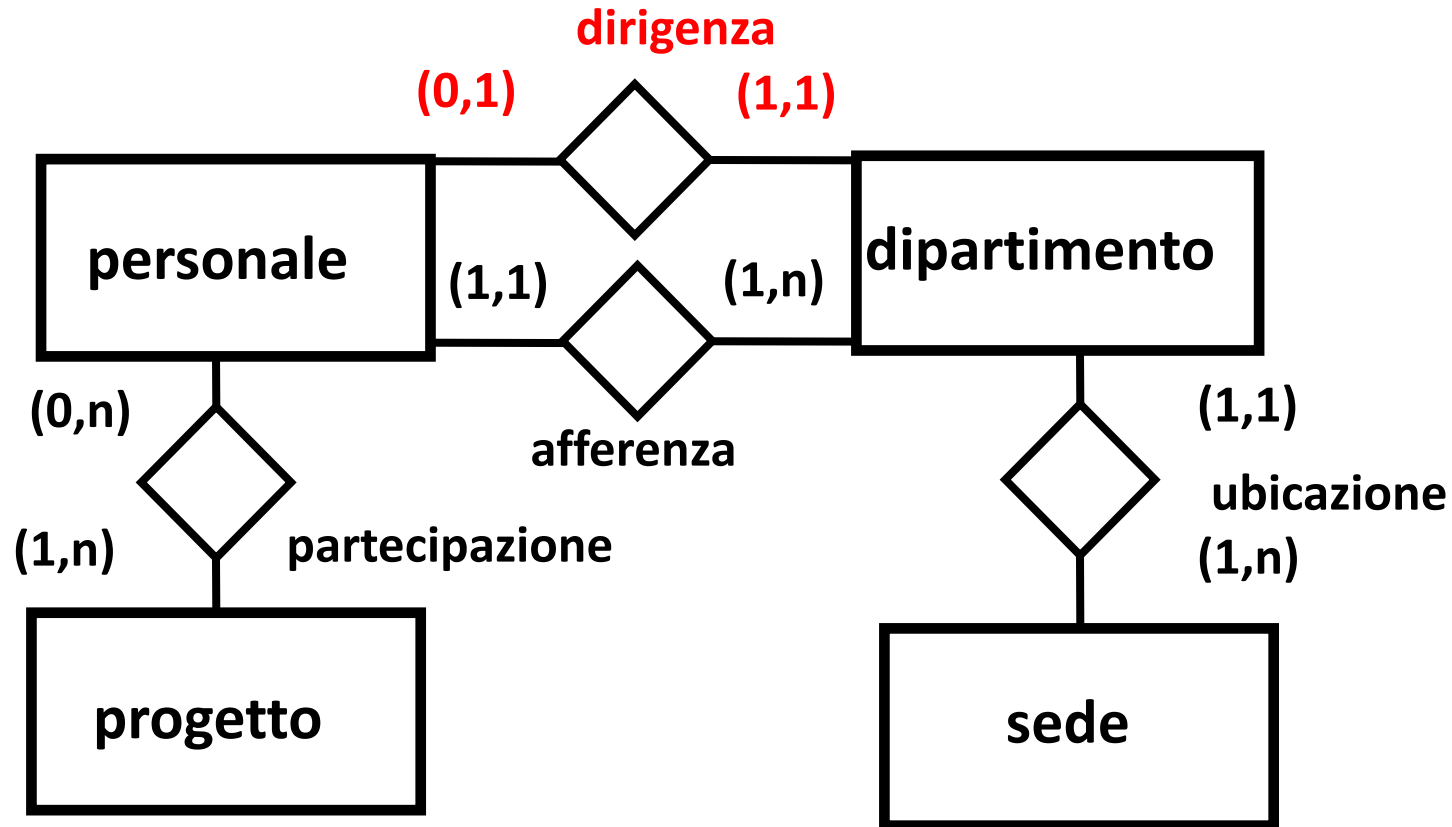
schema iniziale





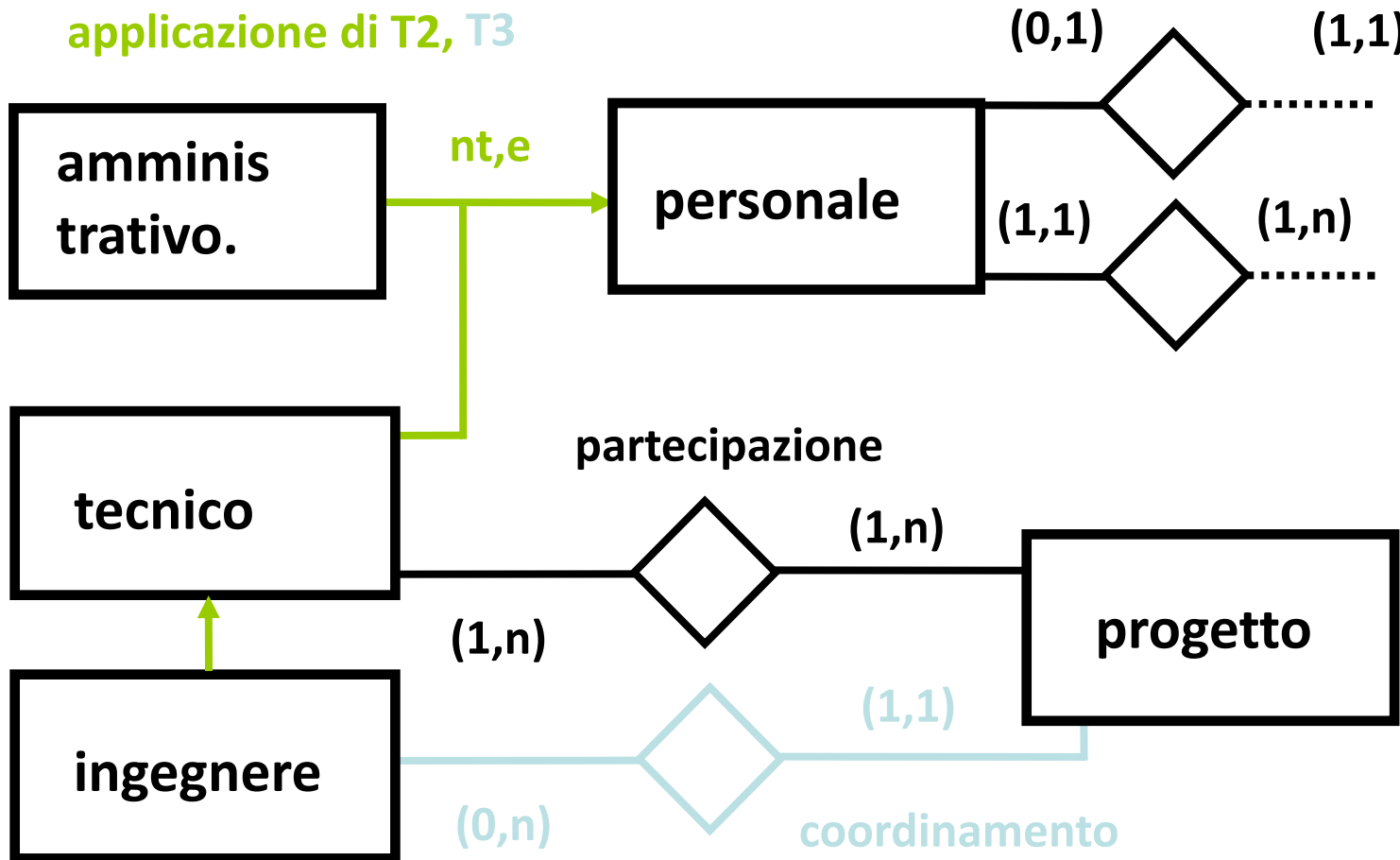
# passo 2

applicazione di T3



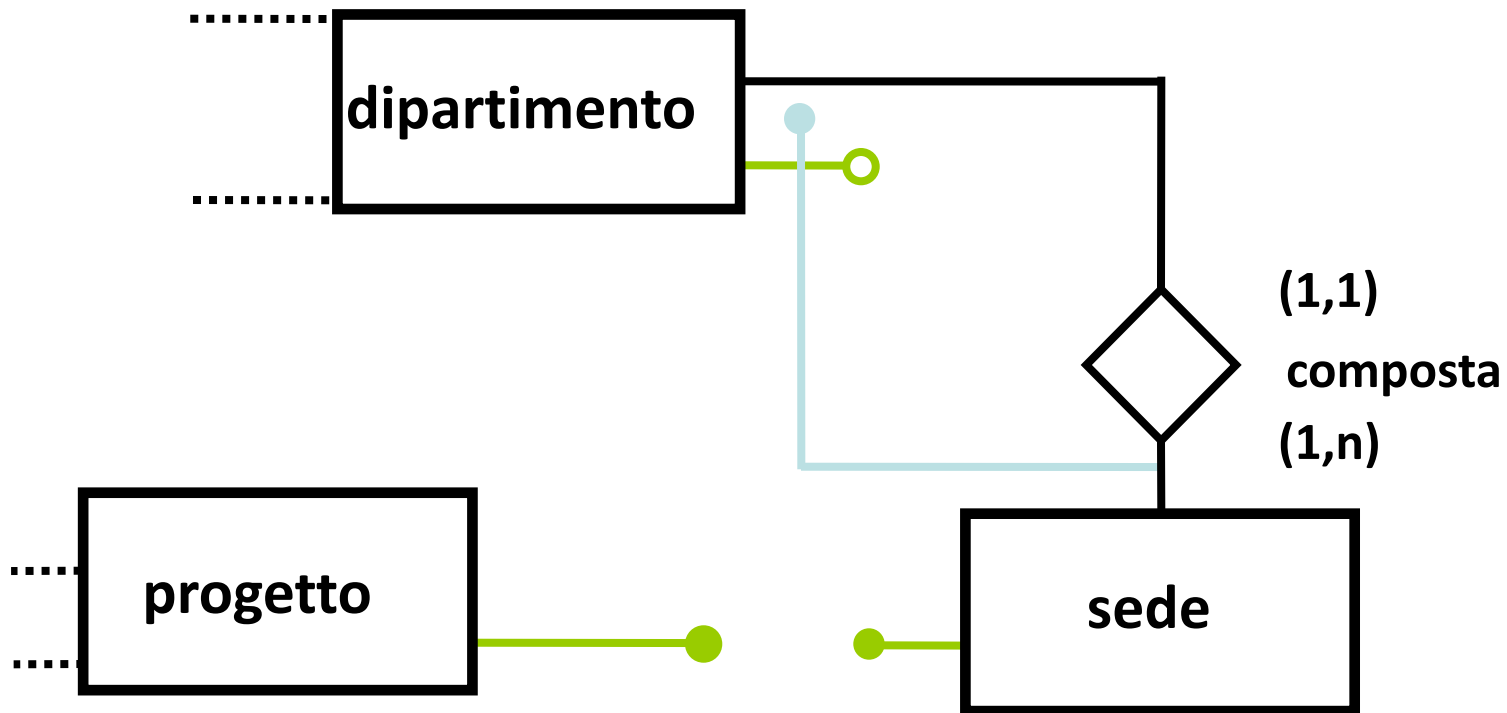
# passo 3

applicazione di T2, T3



# passo 4

applicazione di T5



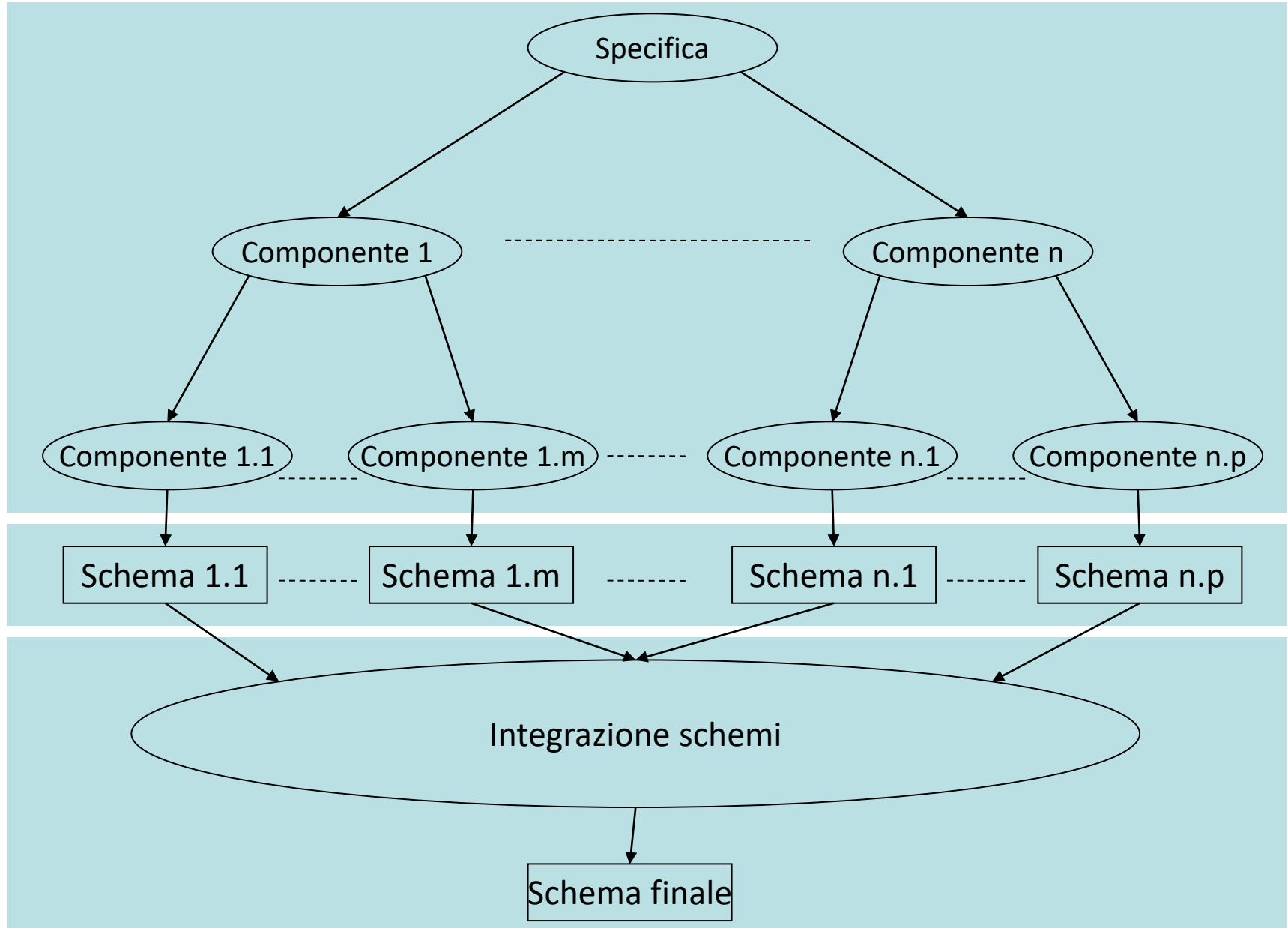
# Valutazione

- vantaggi:
  - il progettista descrive inizialmente lo schema trascurando i dettagli
  - precisa lo schema gradualmente
- problema:
  - non va bene per applicazioni complesse perché è difficile avere una visione globale precisa iniziale di tutte le componenti del sistema

# Strategia bottom-up

- le specifiche nascono suddivise per sottoprogetti descriventi frammenti limitati della realtà da schematizzare
- si sviluppano i sottoschemi separati
- si fondono i sottoschemi per ottenere lo schema finale

# Strategia bottom-up



# Primitive di trasformazione

## Bottom-Up

- **Trasformazione**

- **T1**: si individua nella specifica una classe di oggetti con proprietà comuni e si introduce un'**entità** corrispondente.

concetto

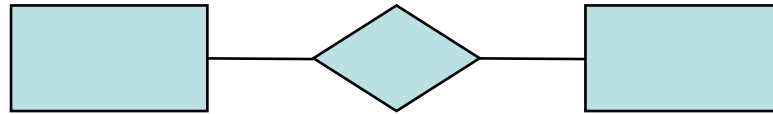


# Primitive di trasformazione

## Bottom-Up

- **Trasformazione**

- **T2**: si individua nella specifica un legame logico fra entità e si introduce una **associazione** fra esse.

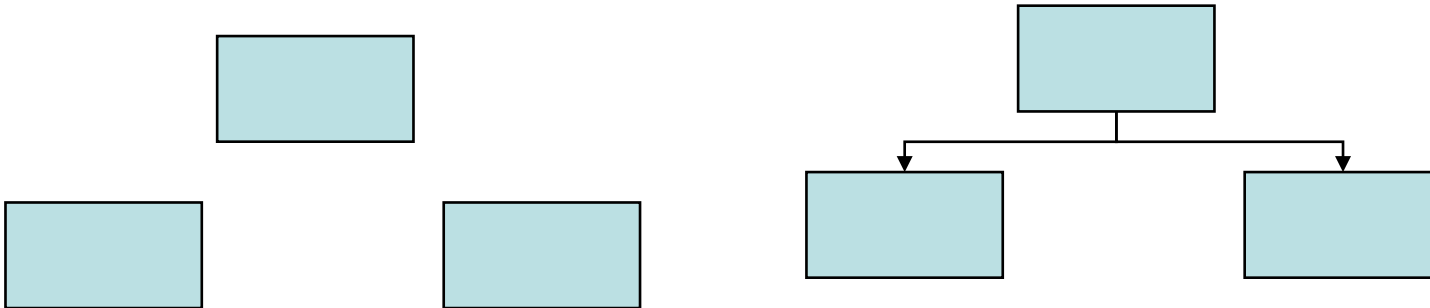




# Primitive di trasformazione

## Bottom-Up

- **Trasformazione**
  - **T3**: si individua una **generalizzazione** fra entità.

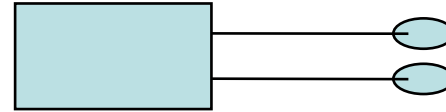
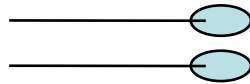


# Primitive di trasformazione

## Bottom-Up

- **Trasformazione**

- **T4**: a partire da una serie di **attributi** si individua un'entità che li aggrega.

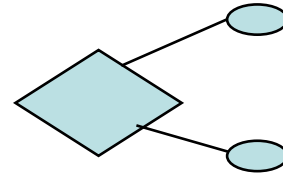
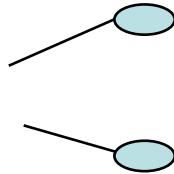


# Primitive di trasformazione

## Bottom-Up

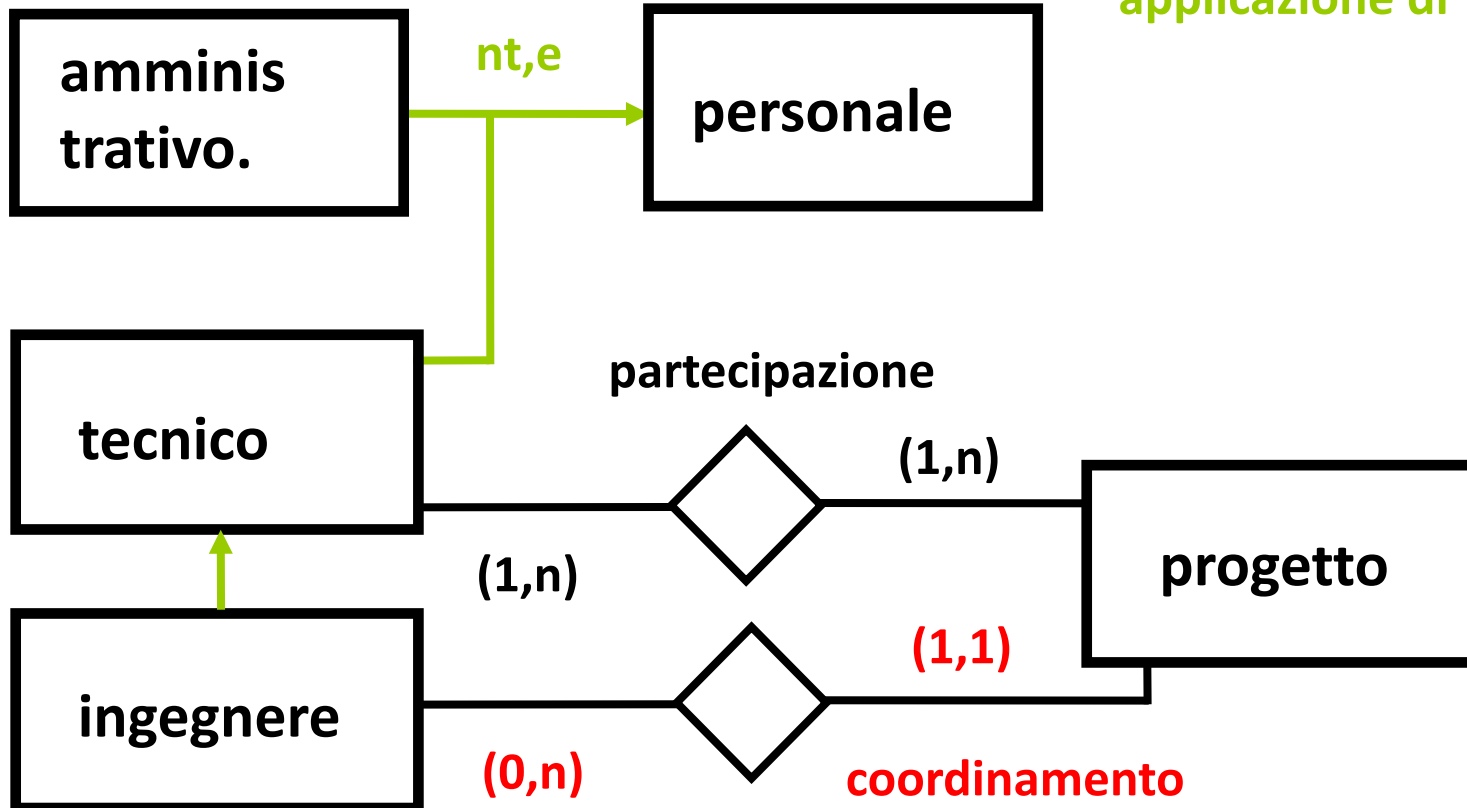
- **Trasformazione**

- **T5**: a partire da una serie di **attributi** si individua una relazione che li aggrega.



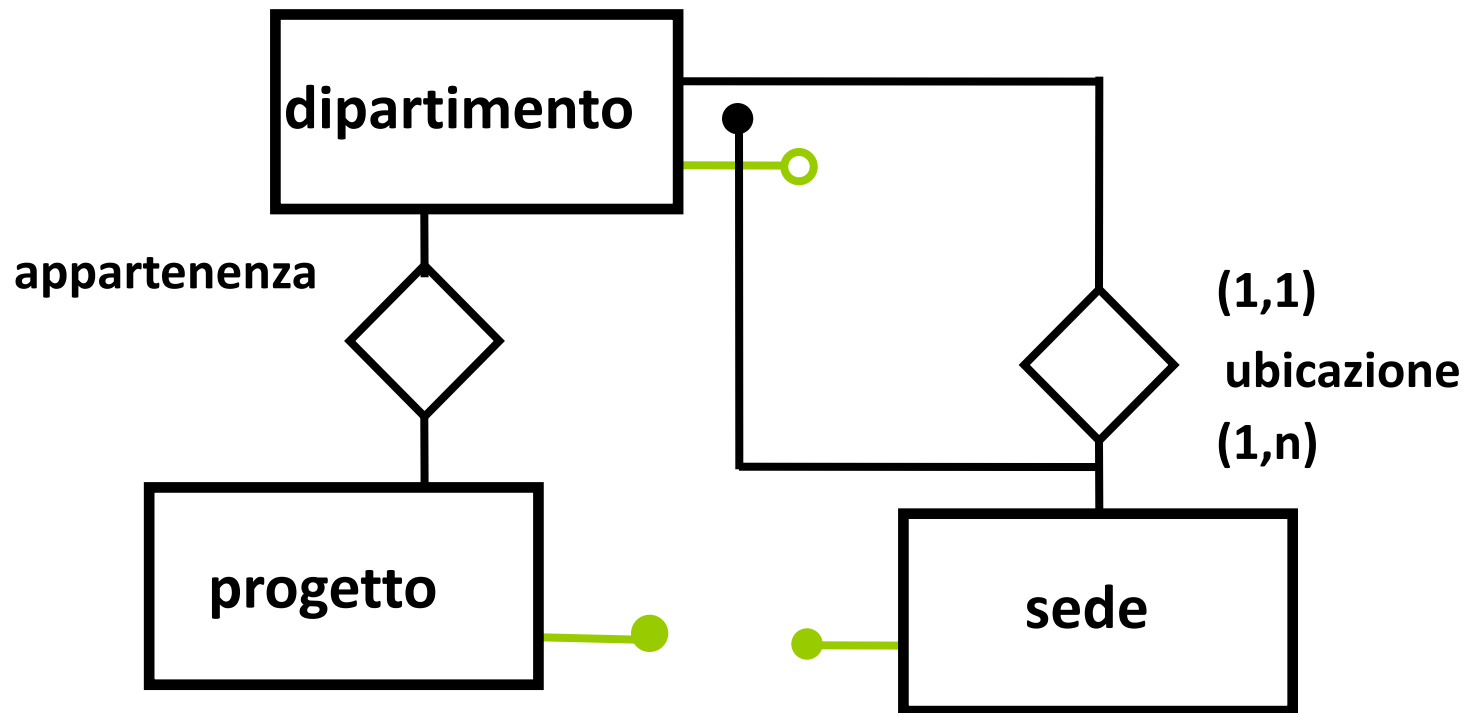
# Sviluppo bottom-up: schema 1

applicazione di T3, T2

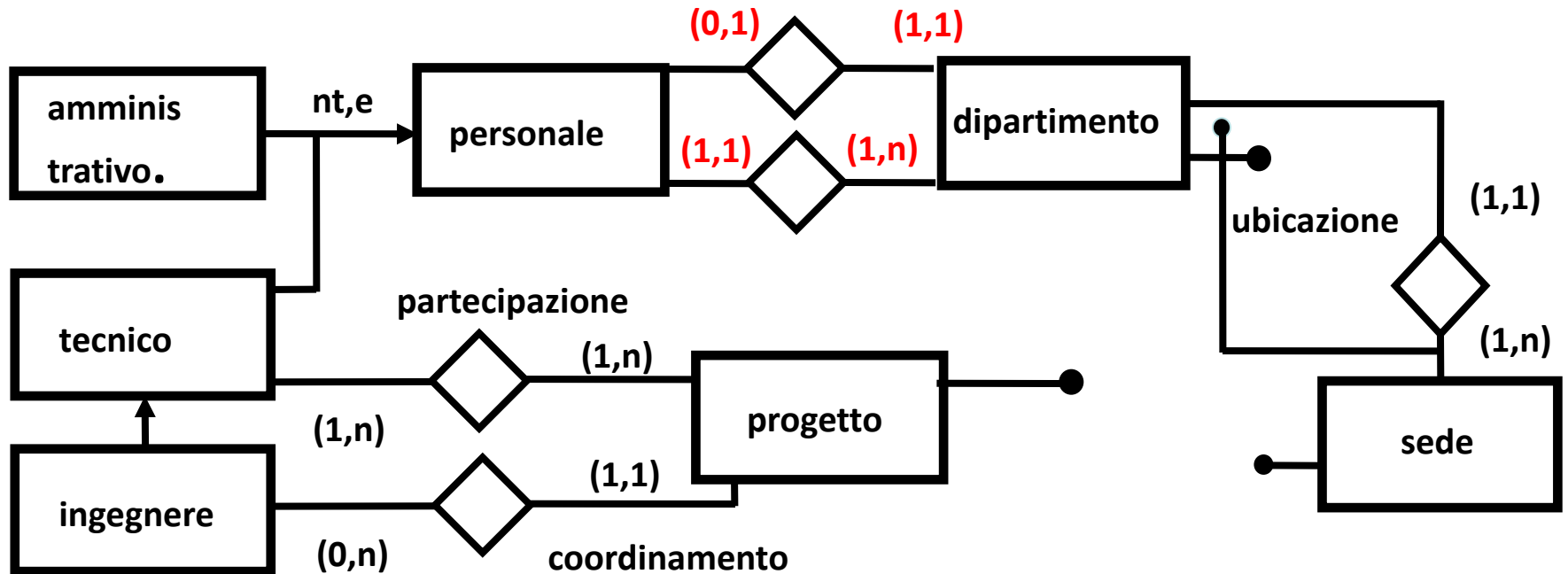


# Sviluppo bottom-up: schema 2

applicazione di T4



# Sviluppo bottom-up: schema integrato



# Vantaggi e Svantaggi della Strategia Bottom-Up

- Si adatta bene ad una progettazione di gruppo in cui , diversi progettisti possono sviluppare parti disgiunte che possono essere assemblate successivamente.
- L'integrazione di sistemi concettualmente diversi comporta notevoli difficoltà.

# Ulteriori strategie

- **inside-out:** è una variante della bottom-up, si sviluppano schemi parziali in aggiunta a sottoschemi già definiti precedentemente e separatamente. Inizialmente si sviluppano alcuni concetti e poi si estendono a macchia d'olio.
- **strategia mista:** cerca di combinare i vantaggi top-down e bottom-up: il progettista divide i requisiti in componenti separate (come nel bottom-up) ma, allo stesso tempo, definisce uno *schema scheletro*, contenente, a livello astratto, i concetti principali dell'applicazione. Questo fornisce una visione unitaria, anche se astratta, dell'intero progetto e può *guidare* le fasi di integrazione dei sottoschemi



# Metodologia Generale

- Analisi requisiti
  - Costruire glossario dei termini
  - Analizzare i requisiti ed eliminare ambiguità
  - Raggruppare i requisiti in insiemi omogenei
- Passo base
  - Individuare i concetti più rilevanti e rappresentarli in uno schema scheletro

# Metodologia Generale

- Passo di decomposizione
  - //Da effettuare se opportuno o necessario
  - Effettuare una decomposizione dei requisiti con riferimento ai concetti presenti nello schema scheletro

# Metodologia Generale

- Passo iterativo

//da ripetere a tutti i sottoschemi (se presenti)  
finché ogni specifica è stata rappresentata

- Raffinare i concetti presenti sulla base delle loro specifiche
- Aggiungere nuovi concetti allo schema per descrivere specifiche non ancora descritte

# Metodologia Generale

- Passo di integrazione
  - //da effettuare se sono presenti diversi sottoschemi
  - Integrare i vari sottoschemi in uno schema generale facendo riferimento allo schema scheletro
- Analisi di qualità

# Qualità di uno Schema Concettuale

- Viene giudicata in base a delle proprietà che lo schema deve possedere:
  - Correttezza
  - Completezza
  - Leggibilità
  - Minimalità

# Correttezza e Completezza

- **Correttezza:** se si utilizzano propriamente i costrutti. Gli errori possono essere **sintattici** : uso non ammesso dei costrutti (ad esempio **generalizzazione fra relazioni**) o **semantici** : uso che non rispetta il loro significato ( si usa **una relazione per descrivere che un'entità è generalizzazione di un'altra**).
- **Completezza:** *tutti i dati di interesse sono rappresentati e tutte le operazioni possono essere eseguite a partire dai concetti dello schema*

# Leggibilità

- Uno schema è **leggibile** quando rappresenta i requisiti in maniera naturale e facilmente comprensibile. Alcune regole:
  - *disporre al centro* i costrutti con più legami
  - usare linee perpendicolari cercando di minimizzare le intersezioni.
  - Disporre i padri di generalizzazioni sopra i figli
- Verificare con gli utenti la leggibilità

# Minimalità

- Uno schema è **minimale** quando tutte le specifiche sono rappresentate una sola volta. Non devono contenere *ridondanze* ovvero concetti deducibili da altri oppure cicli di relazioni e generalizzazioni.
- Una ridondanza a volte può nascere da una scelta precisa di progettazione



# La Progettazione di una basi di dati - Parte III

## Progettazione logica

Prof. Alfredo Pulvirenti  
Prof. Salvatore Alaimo

# Fasi della Progettazione Logica

- **Ristrutturazione dello schema E-R:**
  - e' una fase indipendente dal modello logico e si basa su criteri di *ottimizzazione* dello schema e di successiva *semplificazione*.
- **Traduzione verso il Modello Logico:**
  - fa riferimento ad un modello logico (ad es. relazionale) e puo' includere ulteriore *ottimizzazione* che si basa sul modello logico stesso (es. normalizzazione).

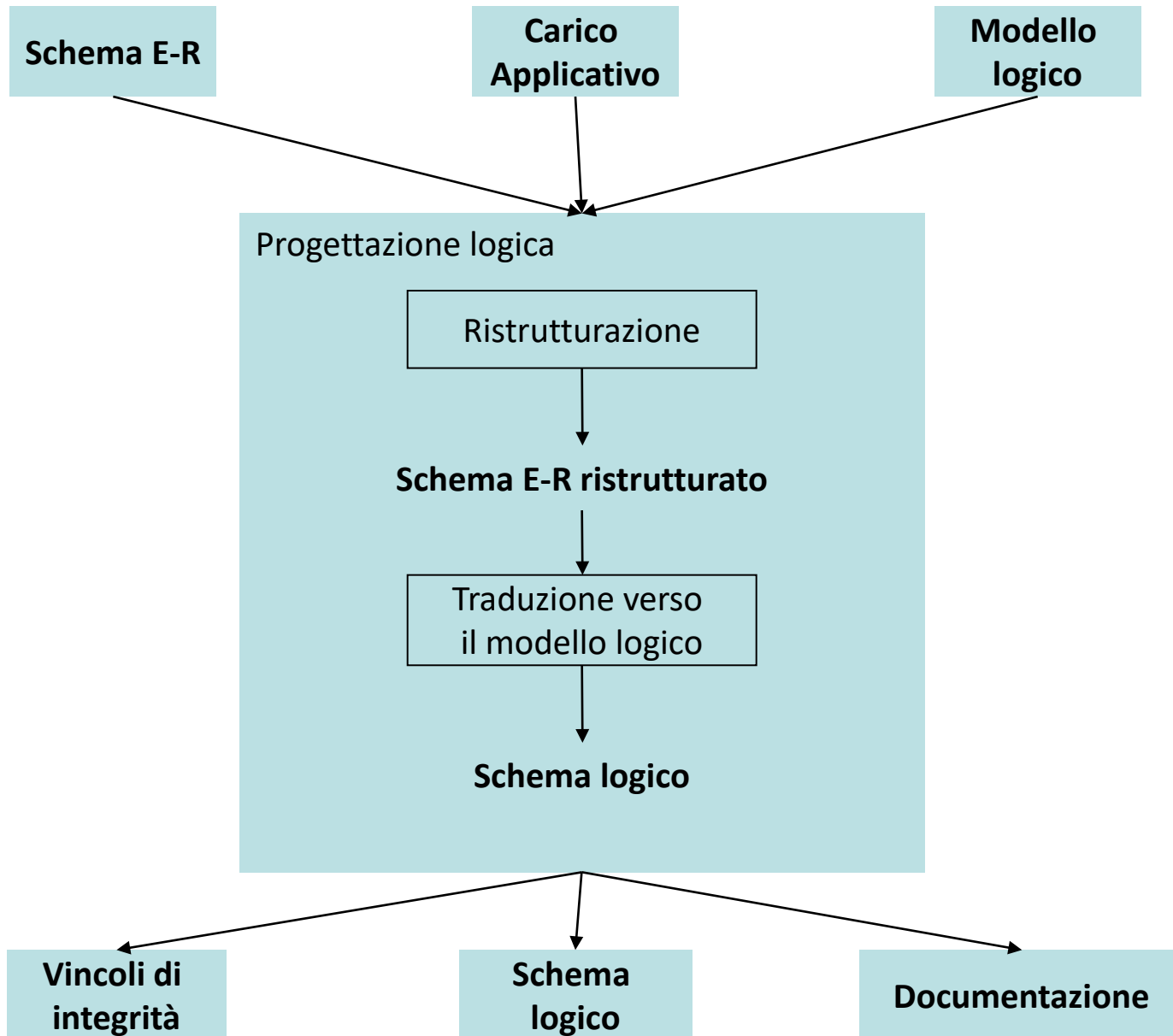
# Input ed output della prima fase

- **Input:**

- *Schema Concettuale E-R iniziale, Carico Applicativo* previsto (in termini di dimensione dei dati e caratteristica delle operazioni)

- **Output :**

- Schema E-R *ristrutturato* che rappresenta i dati e tiene conto degli aspetti realizzativi



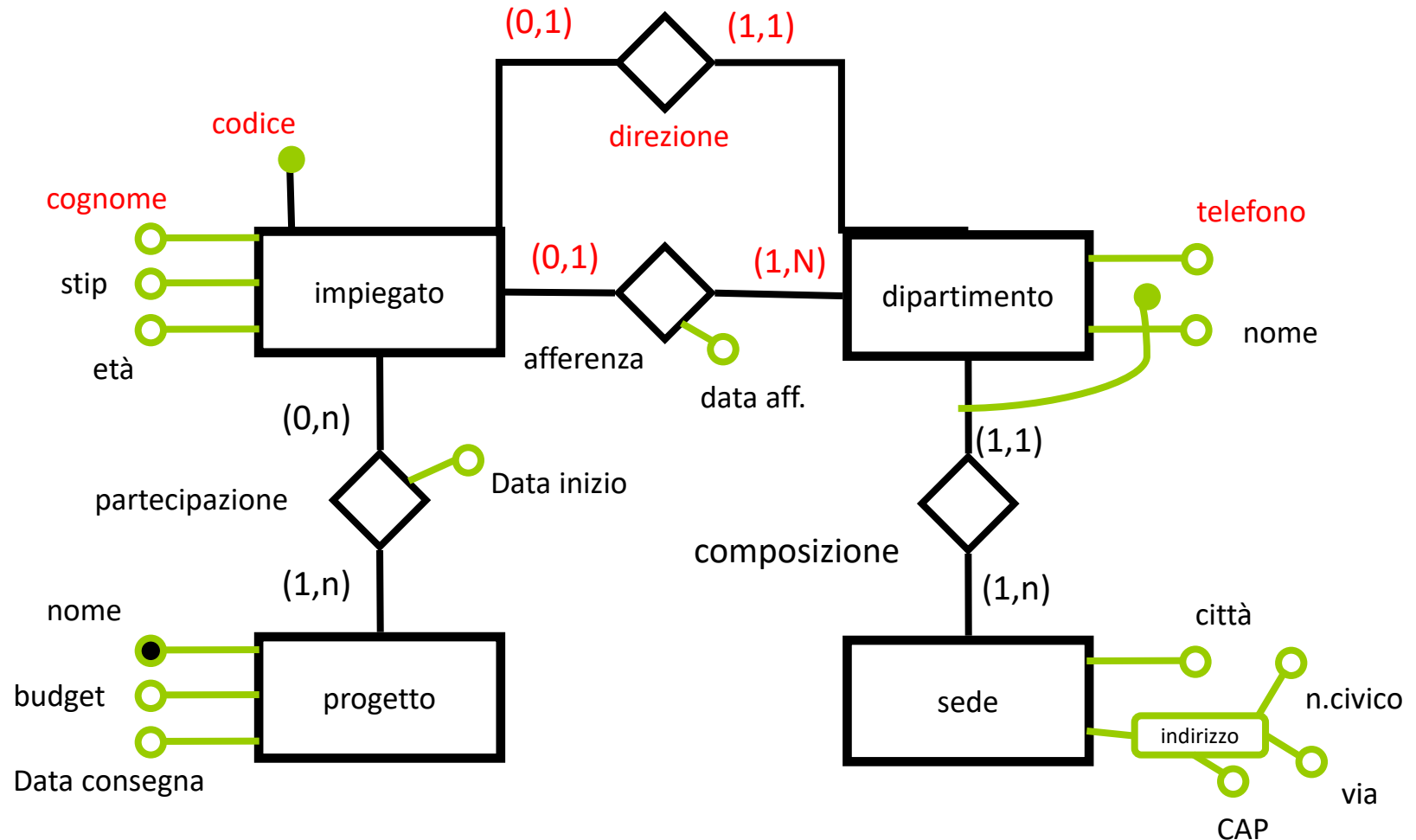
# Analisi delle prestazioni su schemi E-R

- *Indici di prestazione* per la valutazione di schemi E-R sono due:
  - **Costo di un'operazione:** in termini di numero di occorrenze di entità ed associazioni che mediamente vanno visitate per rispondere a quella operazione sulla base di dati (talvolta sarà necessario raffinare questo criterio)
  - **Occupazione di memoria:** viene valutata in termini dello spazio di memoria (misurato in byte) necessario per memorizzare i dati del sistema.

# Volumi e Caratteristiche dei dati

- Per studiare questi due parametri abbiamo bisogno di conoscere:
- **Volume dei dati:**
  - a) numero (medio) di occorrenze di ogni entita' ed associazione
  - b) dimensioni di ciascun attributo
- **Caratteristiche delle operazioni:**
  - a) tipo di operazione (interattiva o batch)
  - b) frequenza (esecuzioni/tempo)
  - c) dati coinvolti (entita' e o associazioni)

# Esempio di analisi: ditta con sedi in città diverse



# Operazioni dell'esempio

- **Operazione 1:** assegna un impiegato ad un progetto
- **Operazione 2:** trova i dati di un impiegato, del dipartimento nel quale lavora e dei progetti in cui e' coinvolto
- **Operazione 3:** trova i dati di tutti gli impiegati di un certo dipartimento
- **Operazione 4:** per ogni sede, trova i dipartimenti con il cognome del direttore e l'elenco degli impiegati.



## TABELLA DEI VOLUMI

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

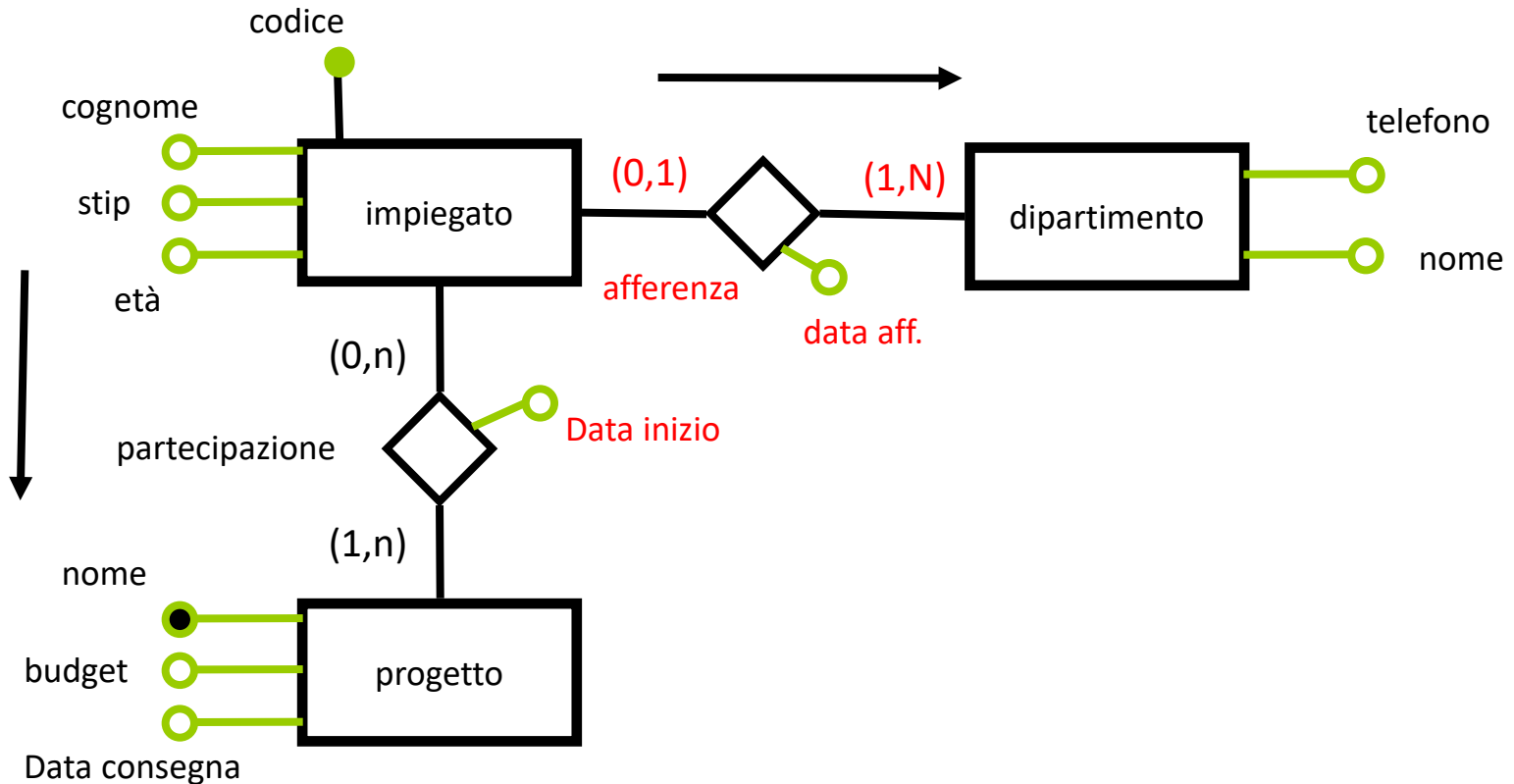
## TABELLA DELLE OPERAZIONI

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

# Stima dei costi

- Avendo a disposizione questi dati è possibile stimare i costi di ogni operazione contando il numero di accessi alle occorrenze di entità e relazioni necessario per eseguire l'operazione.
- Prendiamo per esempio **Operazione 2**: *trova i dati di un impiegato, del dipartimento nel quale lavora e dei progetti in cui e' coinvolto* e facciamo riferimento allo schema di operazione. Si assuma che ogni impiegato partecipa in media a 3 progetti.

# Esempio dell' operazione 2



# Stima del costo dell'operazione 2

- Dobbiamo accedere ad:
  - un'occorrenza di *Impiegato* e di *Afferenza* e quindi di *Dipartimento*;
  - Successivamente, per avere i dati dei progetti a cui lavora, dobbiamo accedere (in media) a tre occorrenze di *Partecipazione* e quindi a tre entità *Progetto*.
  - Tutto viene riassunto nella tavola degli accessi

# Tavola degli accessi

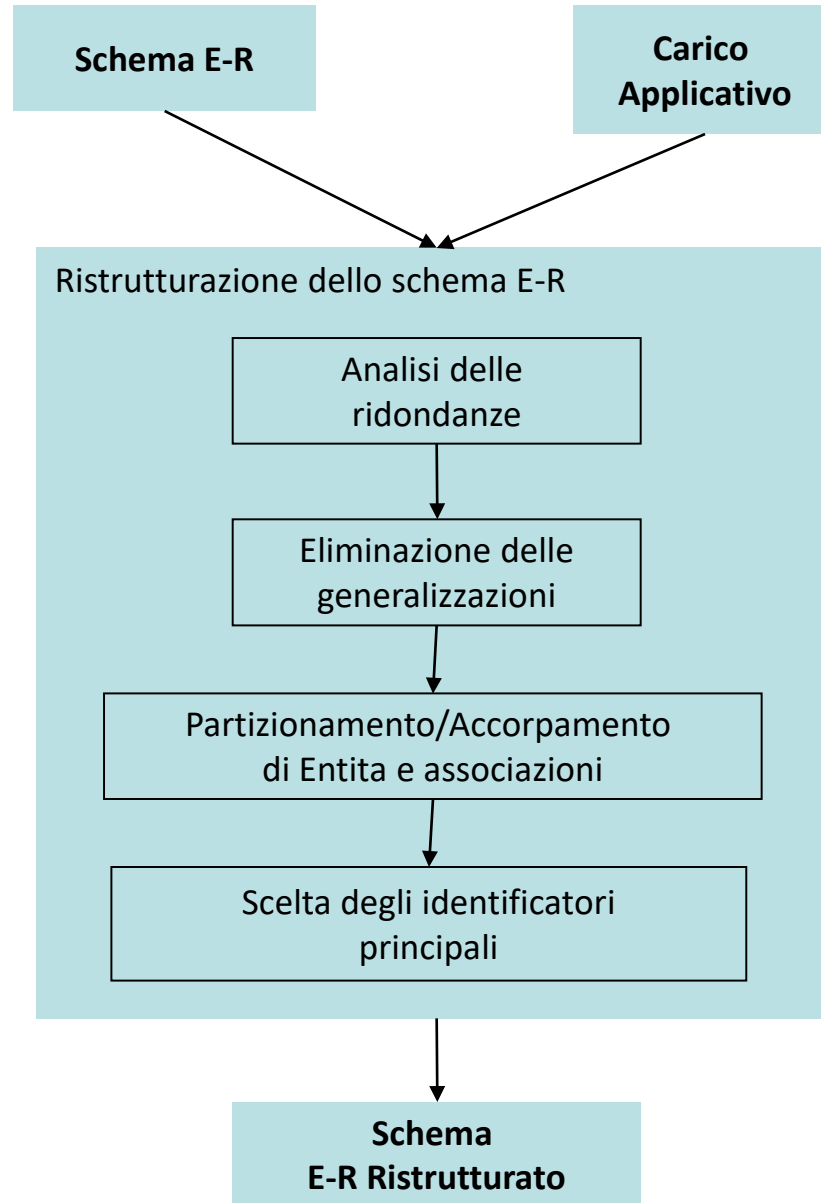
## TAVOLA DEGLI ACCESSI

CONCETTO	COSTRUTTO	ACCESSI	TIPO
Impiegato	Entita'	1	L
Afferenza	Relazione	1	L
Dipartimento	Entita'	1	L
Partecipazione	Relazione	3	L
Progetto	Entita'	3	L

L lettura, S scrittura. In genere la scrittura e' piu' onerosa della lettura (1S = 2L)

# Ristrutturazione di schemi E-R

- **Analisi delle Ridondanze:** si decide se eliminare o no eventuali ridondanze.
- **Eliminazione delle Generalizzazioni:** tutte le generalizzazioni vengono analizzate e sostituite da altro.
- **Partizionamento/Accorpamento di entita' ed associazioni:** si decide se partizionare concetti in piu' parti o viceversa accorpare.
- **Scelta degli identificatori primari:** si sceglie un identificatore per quelle entita' che ne hanno piu' di uno



# Analisi delle Ridondanze

- *Attributi derivabili da altri attributi della stessa entità (fattura: importo lordo)*
- *Attributi derivabili da attributi di altre entità (o associazioni) (Acquisto: Importo totale da Prezzo )*
- *Attributi derivabili da operazioni di conteggio (Città: Numero abitanti contando il numero di Residenza )*
- *Associazioni derivabili dalla composizione di altre associazioni in presenza di cicli. Tuttavia i cicli non necessariamente generano ridondanze.*

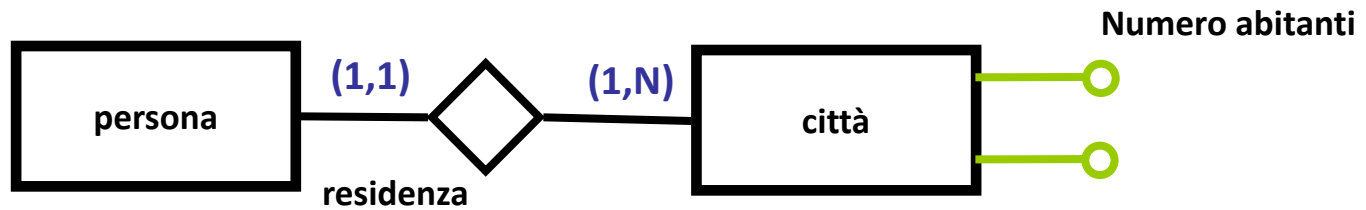


# Dato derivabile

- *Vantaggi*: riduce gli accessi per calcolare il dato derivato.
- *Svantaggi*: occupazione di memoria e necessita' di effettuare operazioni aggiuntive per mantenere il dato aggiornato.
- Decisione: mantenere o eliminare?
  - Basta confrontare i costi di esecuzione delle operazioni sull'oggetto

# Esempio

- Consideriamo l'esempio Città-Persona per l'anagrafica di una regione.
  - *Operazione 1*: memorizza una persona nuova con la relativa città.
  - *Operazione 2*: stampa tutti i dati di una città (incluso il numero di abitanti).
- Valutiamo gli indici di prestazione per l'attributo Numero Abitanti



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

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

# Valutazione in presenza della ridondanza

- Costo memoria per l'attributo NUMABITANTI:  
Assumendo che il numero di abitanti richieda 4 byte il dato richiede  $4 \times 200 = 800$  byte.
  - Operazione 1 richiede
    - un accesso in scrittura a Persona
    - uno in scrittura a Residenza
    - uno in lettura per cercare la città'
    - ed uno in scrittura (per incrementare il numero di abitanti) a Città
      - ripetuto 500 volte
      - » si hanno 1500 accessi in scrittura e 500 in lettura.
  - L'operazione 2 richiede
    - un solo accesso in lettura a Città
      - 2 volte al giorno. (trascurabile...)
  - Supponendo che la scrittura ha un costo doppio rispetto ad una lettura si hanno 3500 accessi al giorno in presenza della ridondanza.

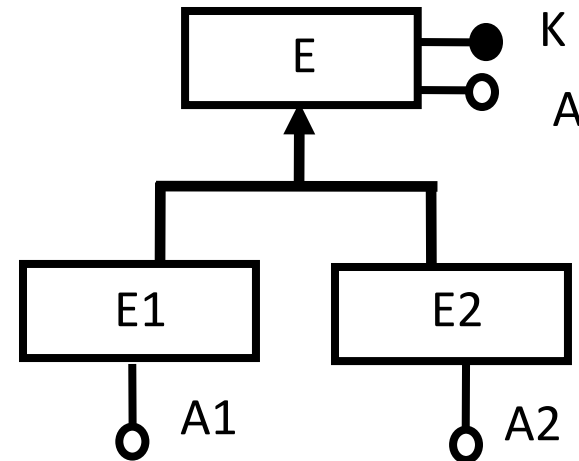
# Valutazione in assenza della ridondanza

- Per l'operazione 1,
  - un accesso in scrittura a Persona
  - ed uno in scrittura a Residenza
    - Ripetuto 500 volte
      - un totale di 1000 accessi in scrittura al giorno.
- Per l'operazione 2 abbiamo bisogno di un accesso
  - in lettura a Città (possiamo trascurare) e
  - 5000 accessi in lettura a Residenza in media (persone/città)
    - Ripetuto 2 volte al giorno
      - per un totale di 10.000 accessi in lettura al giorno.
- Considerando doppi gli accessi il scrittura, il totale e' di 12000. Quindi 8500 in più rispetto al caso di ridondanza contro meno di un solo Kilobyte di memoria in più.

# Eliminazione delle gerarchie

il modello relazionale non rappresenta le gerarchie, le gerarchie sono sostituite da entità e associazioni:

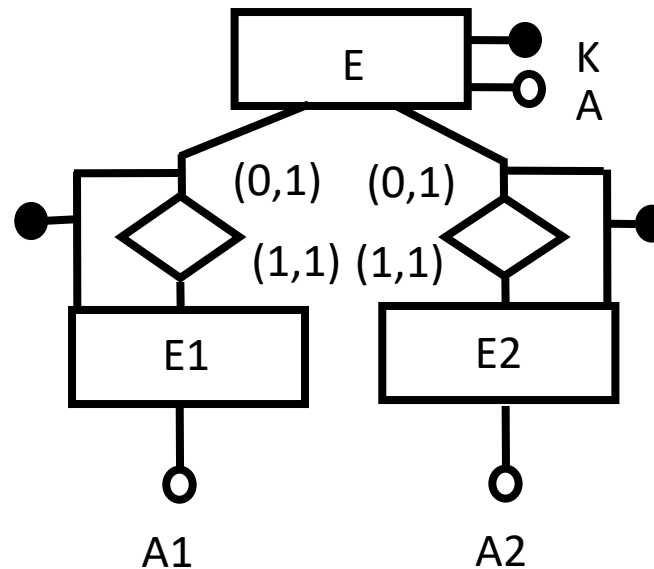
- 1) mantenimento delle entità con associazioni
- 2) collasso verso l'alto
- 3) collasso verso il basso



l'applicabilità e la convenienza delle soluzioni dipendono dalle proprietà di copertura e dalle operazioni previste

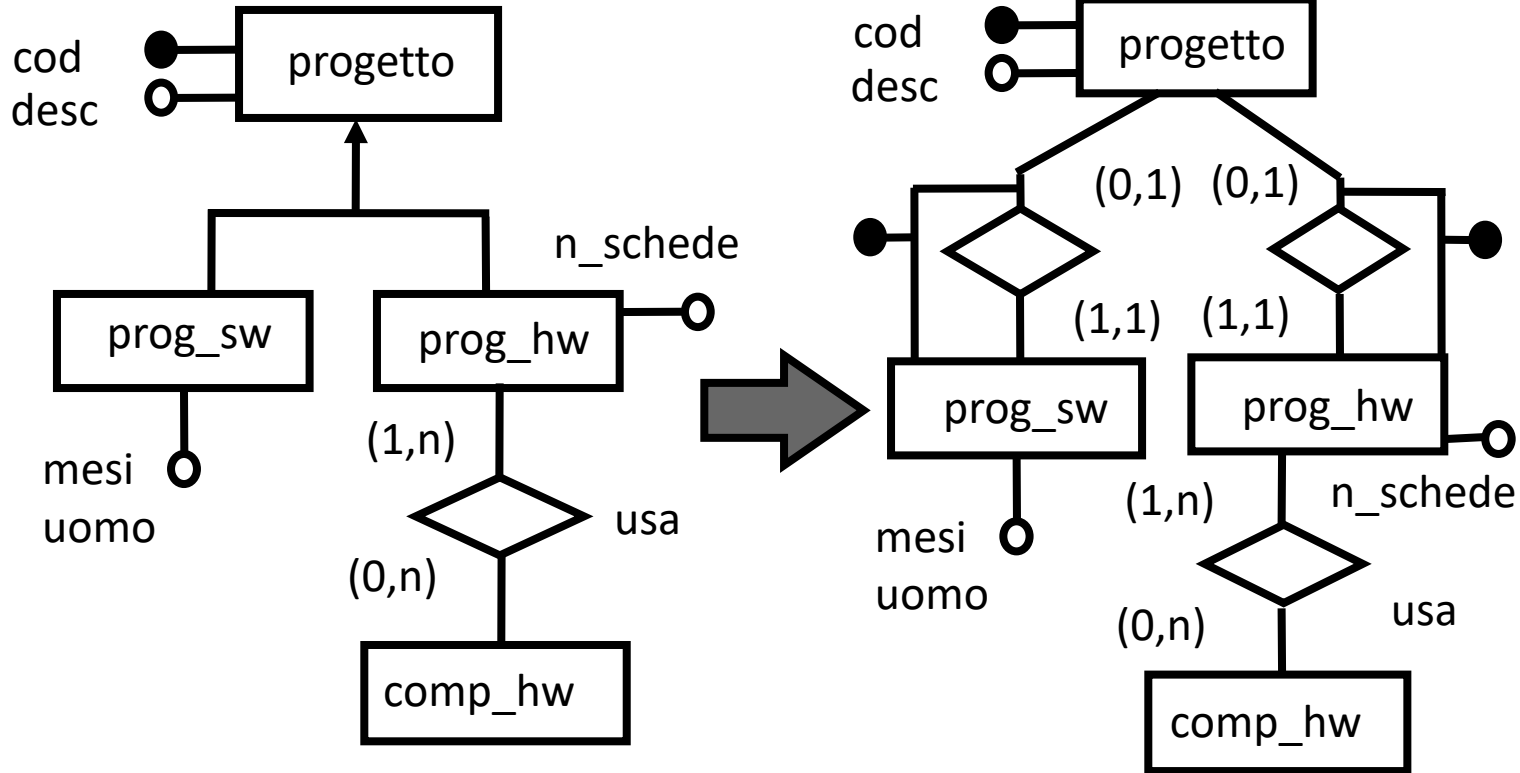
# mantenimento delle entità

- tutte le entità vengono mantenute
- le entità figlie sono in associazione con l'entità padre
- le entità figlie sono identificate esternamente tramite l'associazione



questa soluzione è sempre possibile,  
indipendentemente dalla copertura

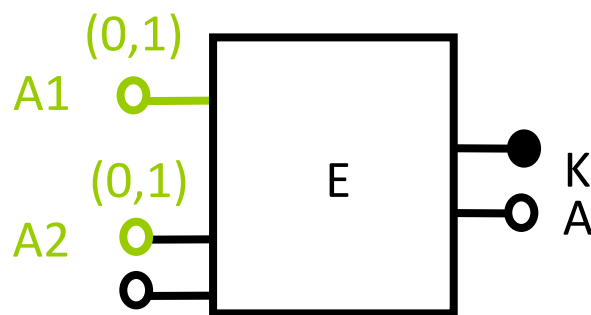
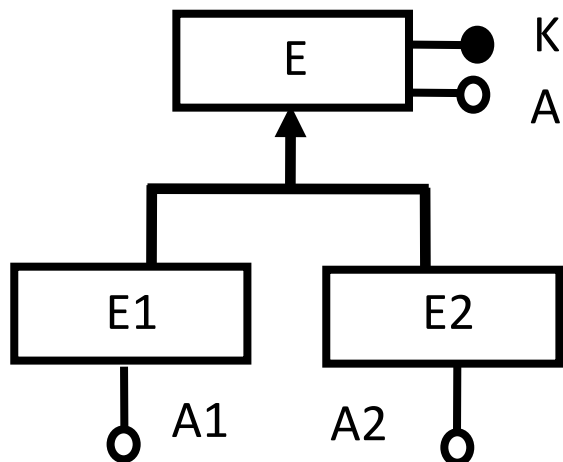
# mantenimento entità - es.:





# eliminazione delle gerarchie

- Il **collasso verso l'alto** riunisce tutte le entità figlie nell'entità padre



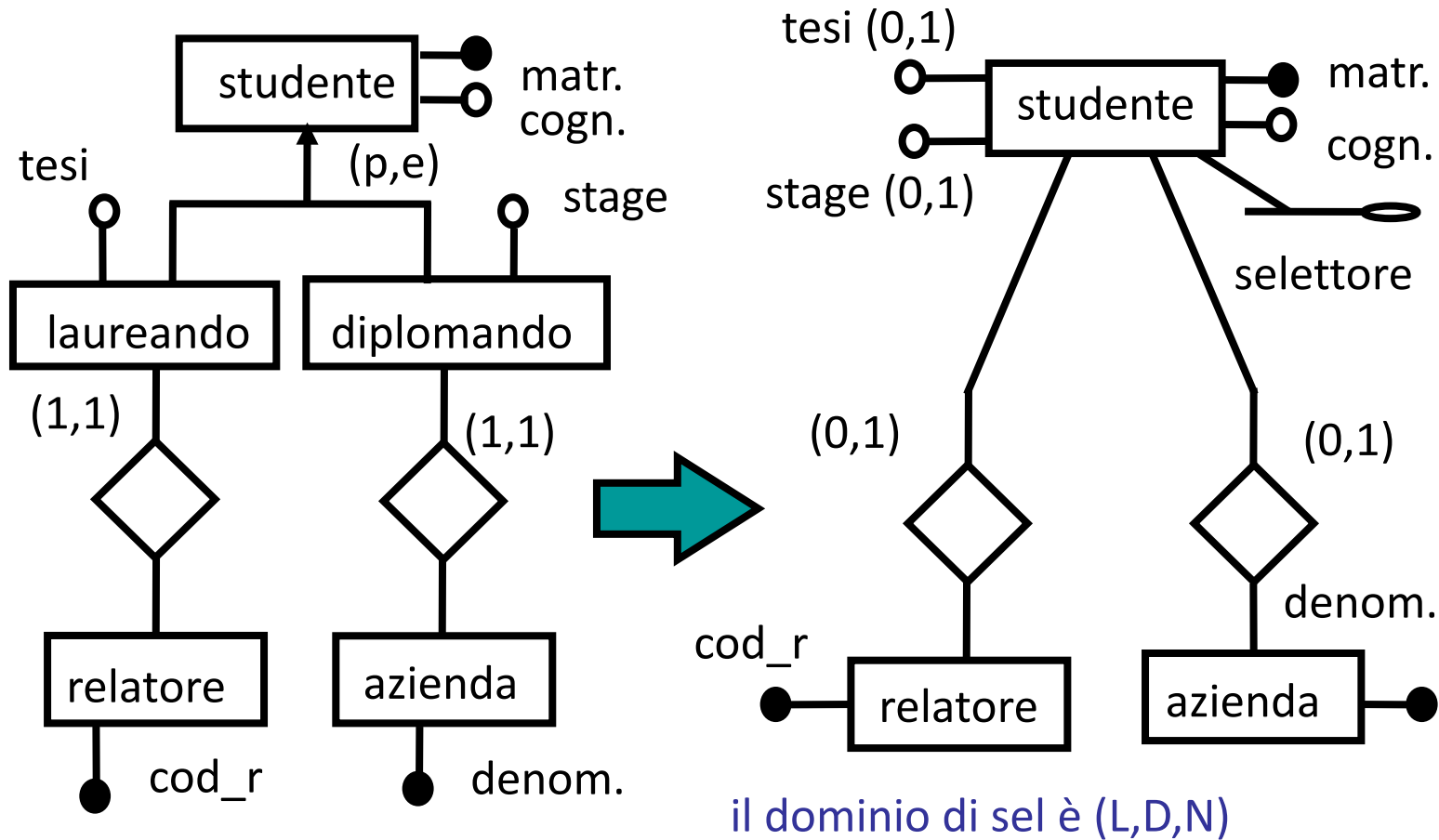
selettore

**selettore** è un attributo che specifica se una istanza di E appartiene a una delle sottoentità

# ISA: collasso verso l'alto

- Il collasso verso l'alto favorisce operazioni che consultano insieme gli attributi dell'entità padre e quelli di una entità figlia:
  - in questo caso si accede a una sola entità, anziché a due attraverso una associazione
- gli attributi obbligatori per le entità figlie divengono opzionali per il padre
  - si avrà una certa percentuale di valori nulli

# ISA: collasso verso l'alto



# ISA: collasso verso il basso

- Collasso verso il basso:
- si elimina l'entità padre trasferendone gli attributi su tutte le entità figlie
  - una associazione del padre è replicata, tante volte quante sono le entità figlie
  - la soluzione è interessante in presenza di molti attributi di specializzazione (con il collasso verso l'alto si avrebbe un eccesso di valori nulli)
  - favorisce le operazioni in cui si accede separatamente alle entità figlie

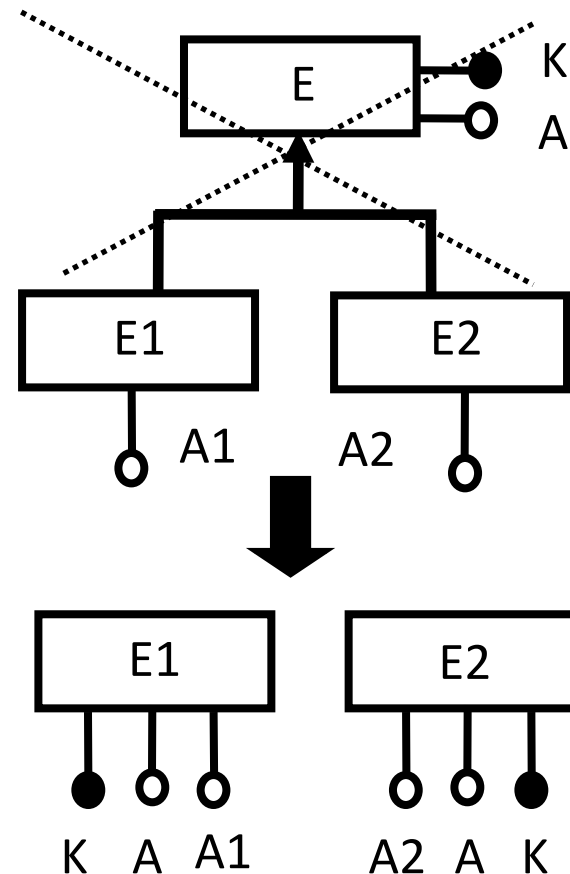
# ISA: collasso verso il basso

limiti di applicabilità:

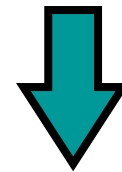
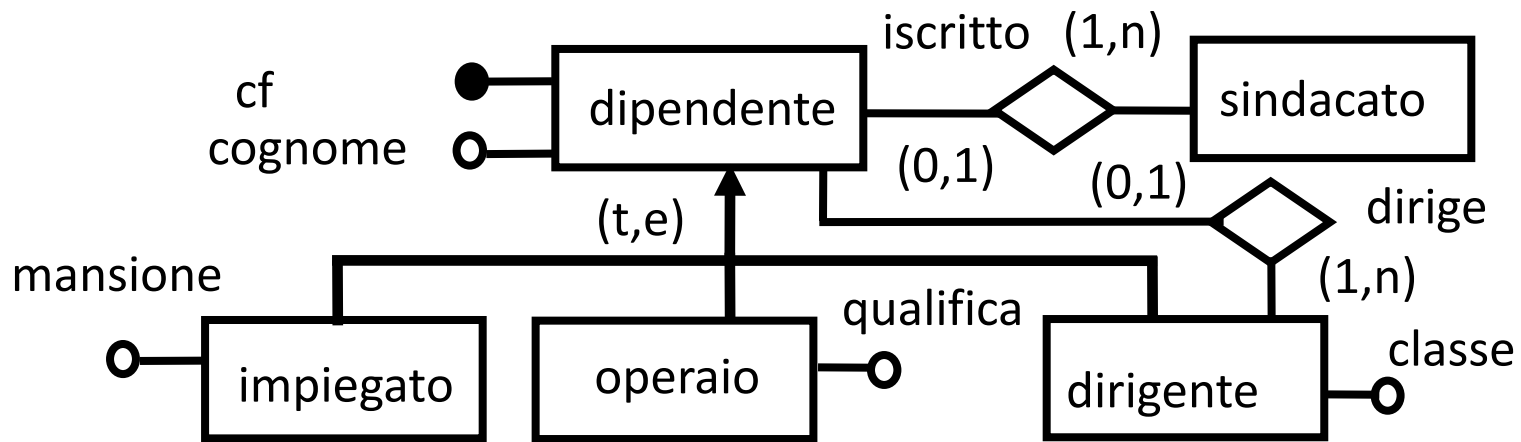
- se la copertura non è totale  
non si può fare:

dove mettere gli E che non sono né E1, né E2 ?

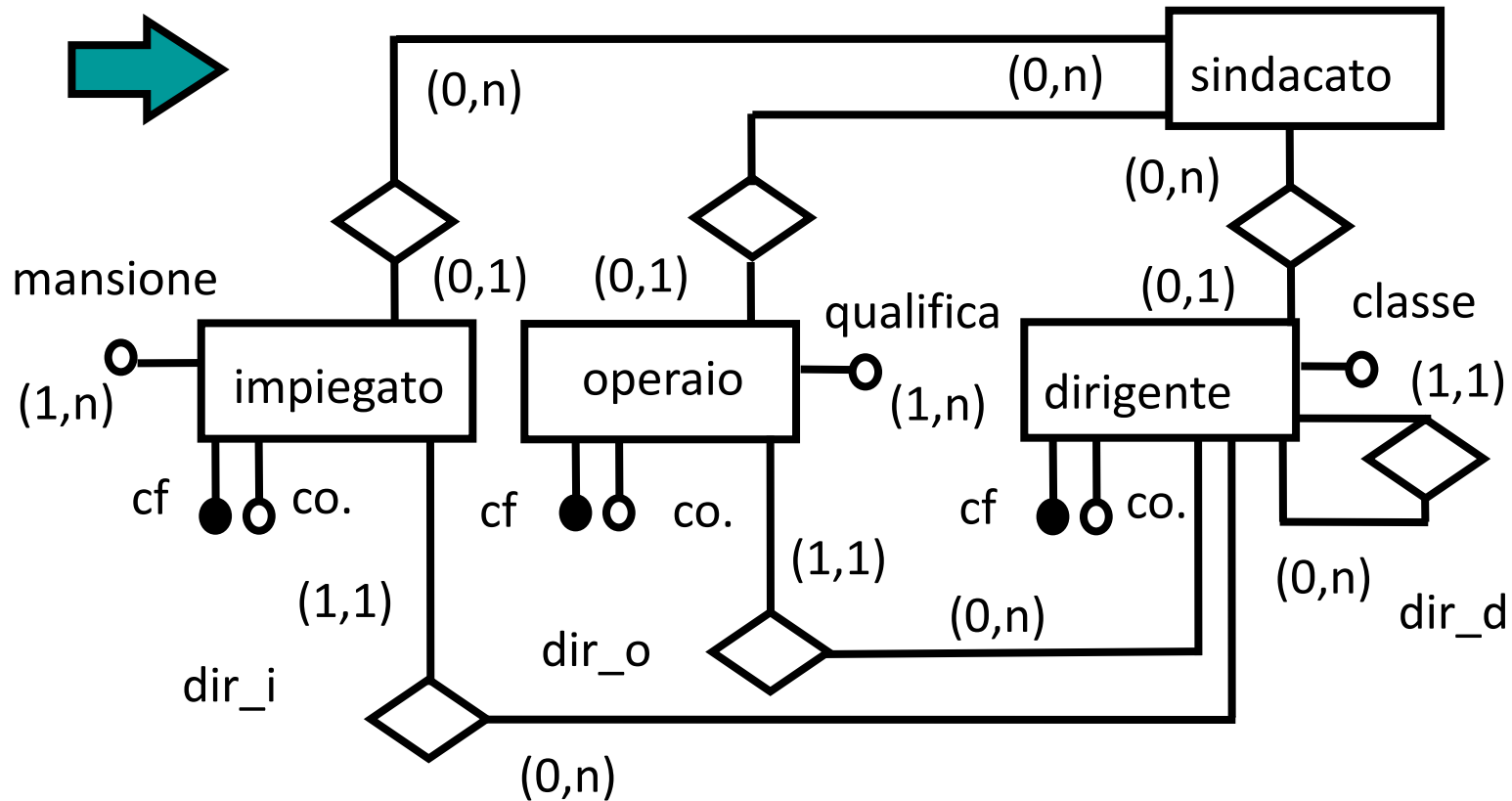
- se la copertura non è esclusiva introduce  
ridondanza: per una istanza presente sia in E1 che in E2 si rappresentano due volte gli attributi di E



collasso verso il basso: es.



collasso verso il basso: es.

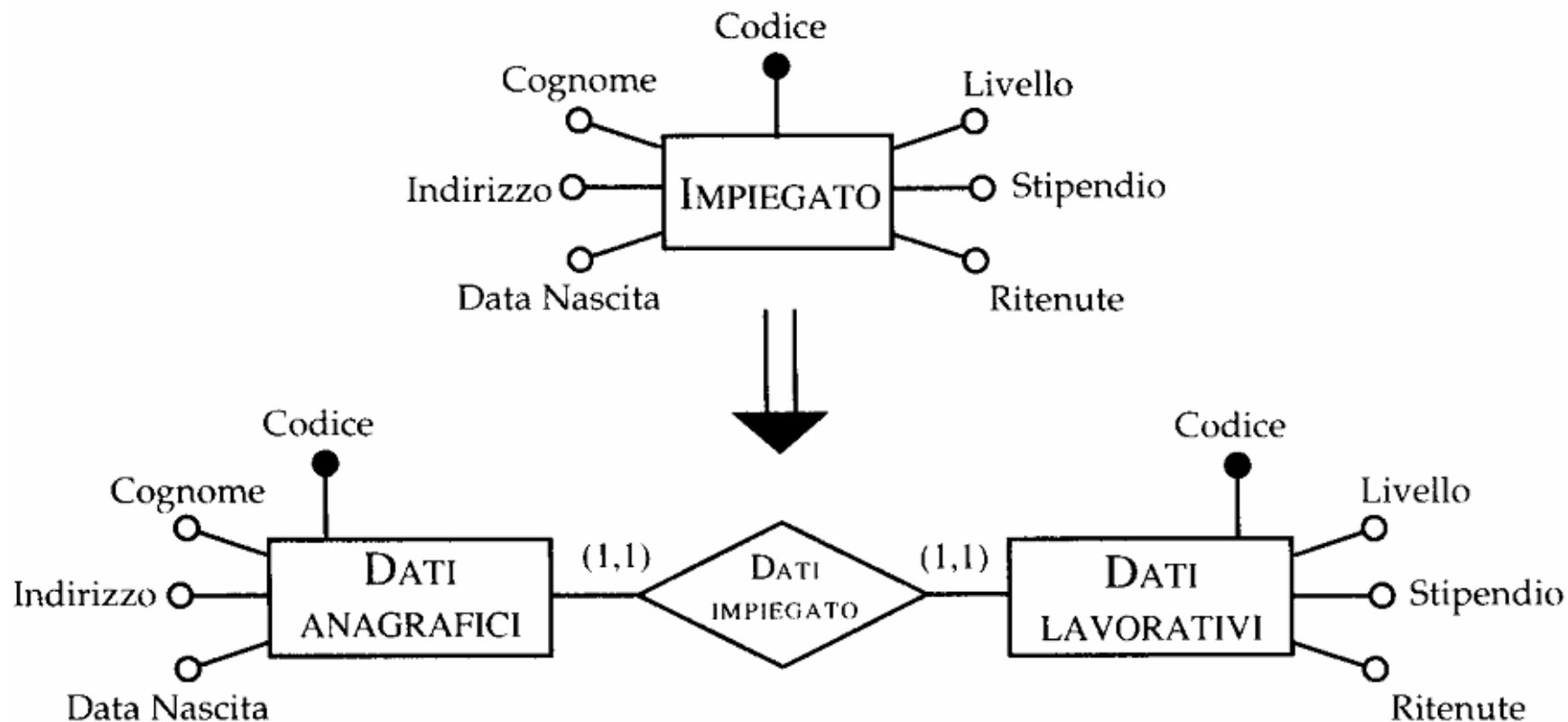


# Partizionamento/Accorpamento

- Il principio generale è il seguente: gli accessi si riducono
  - separando attributi di uno stesso concetto che vengono acceduti da operazioni diverse
  - raggruppando attributi di concetti diversi che vengono acceduti dalle medesime operazioni



# Partizionamento di entità

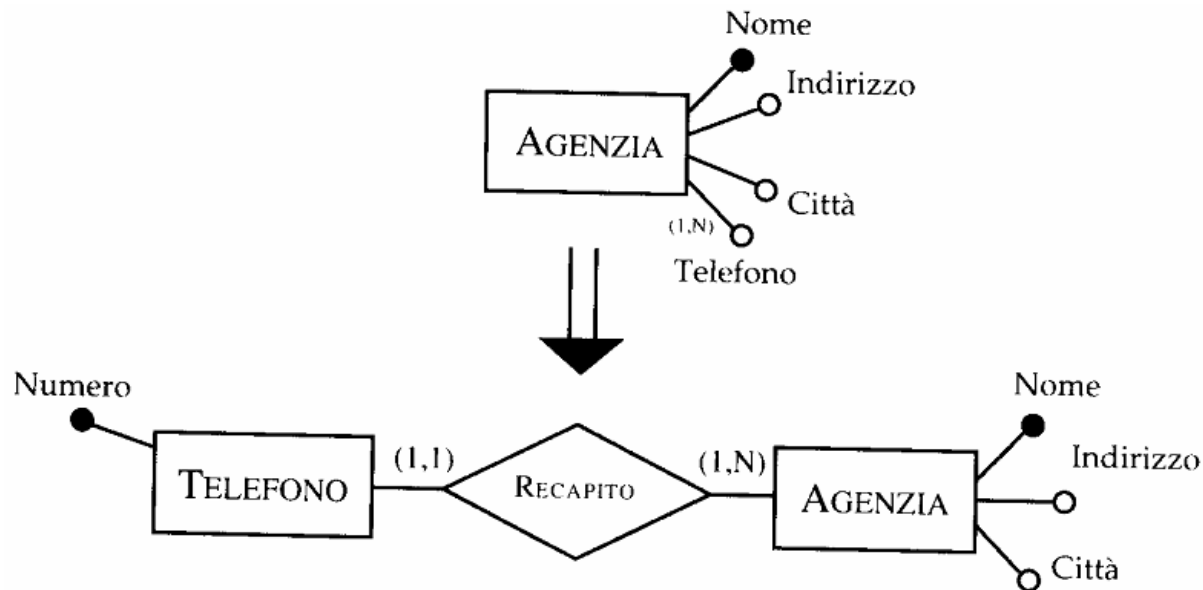


# Partizionamento verticale ed orizzontale

- Nell'esempio precedente vengono create due entità e gli attributi vengono divisi:  
*partizionamento verticale*
- Se invece si suddivide in due entità con gli stessi attributi (ad esempio Analista e Venditore) con operazioni distinte sulle due si ha il *partizionamento orizzontale*

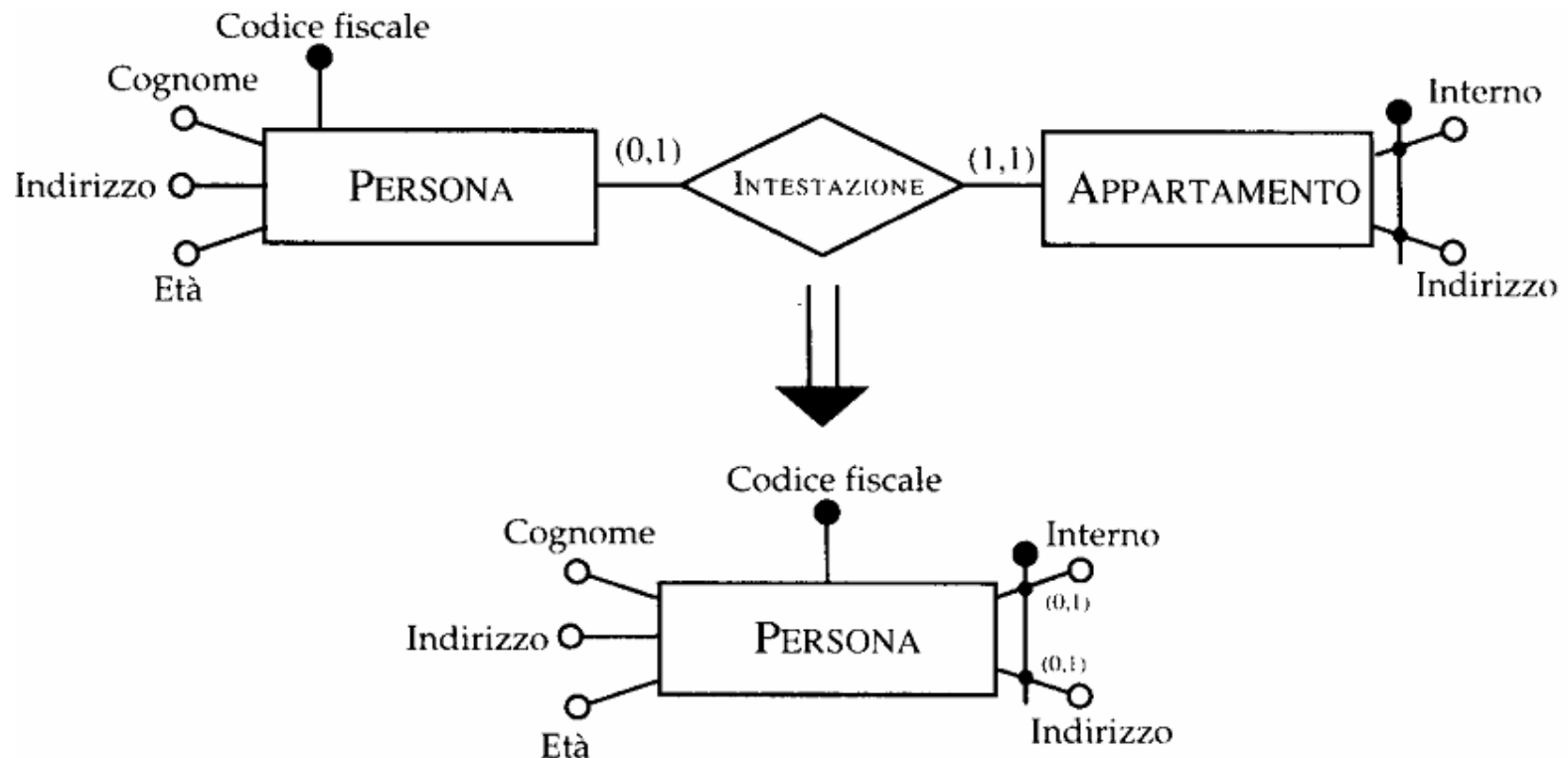
# Eliminazione di attributi multivalore

- Il modello relazionale non li supporta (anche se alcuni sistemi li ammettono)



# Accorpamento di entità

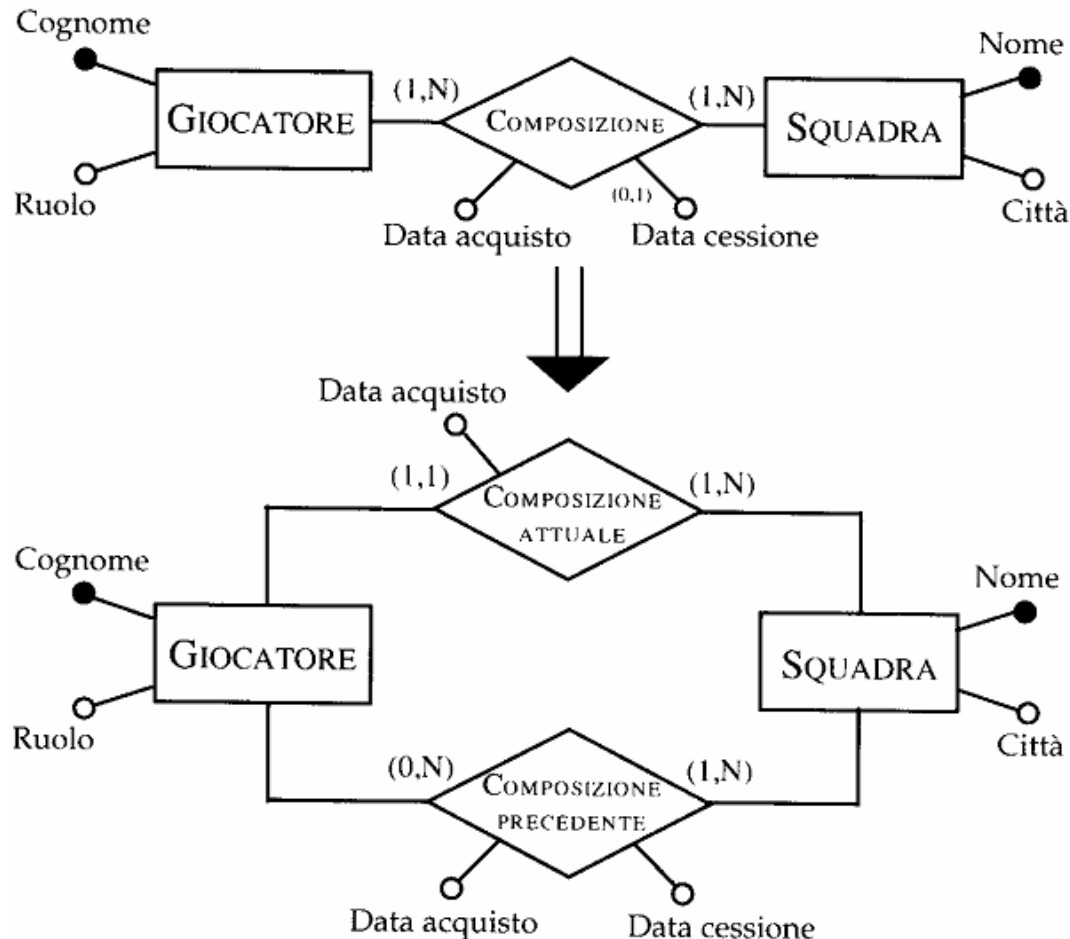
- E' l'operazione inversa del partizionamento



# Quando si fa un accorpamento

- L'accorpamento precedente è giustificato se le operazioni più frequenti su Persona richiedono sempre i dati relativi all'appartamento e quindi vogliamo risparmiare gli accessi alla relazione che li lega.
- *Normalmente gli accorpamenti si fanno su relazioni uno ad uno, raramente su uno a molti mai su molti a molti.*

# Partizionamento/Accorpamento di associazioni

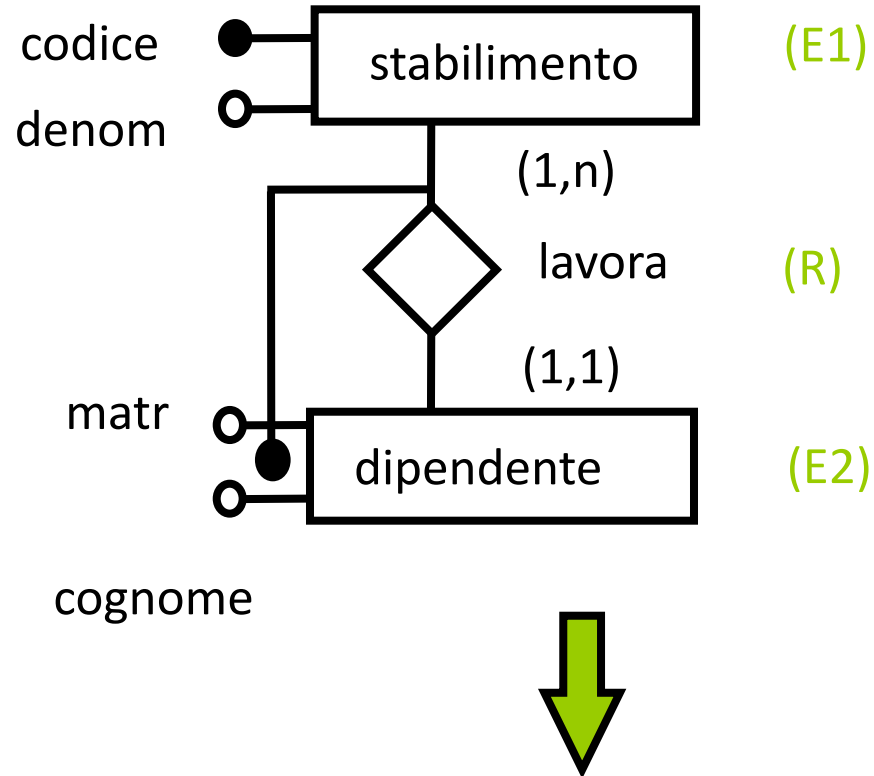


# Scelta della chiave primaria

- È necessario che tra i diversi identificatori di una entità venga designata una chiave primaria:
  - per la chiave primaria occorrerà, infatti, che il DBMS sia provvisto di strumenti per garantire l'unicità dei valori
- criteri euristici di scelta:
  - primo: scegliere la chiave che è usata più frequentemente per accedere all'entità
  - secondo: si preferiscono chiavi semplici a chiavi composte, interne anziché esterne

# identificatori esterni

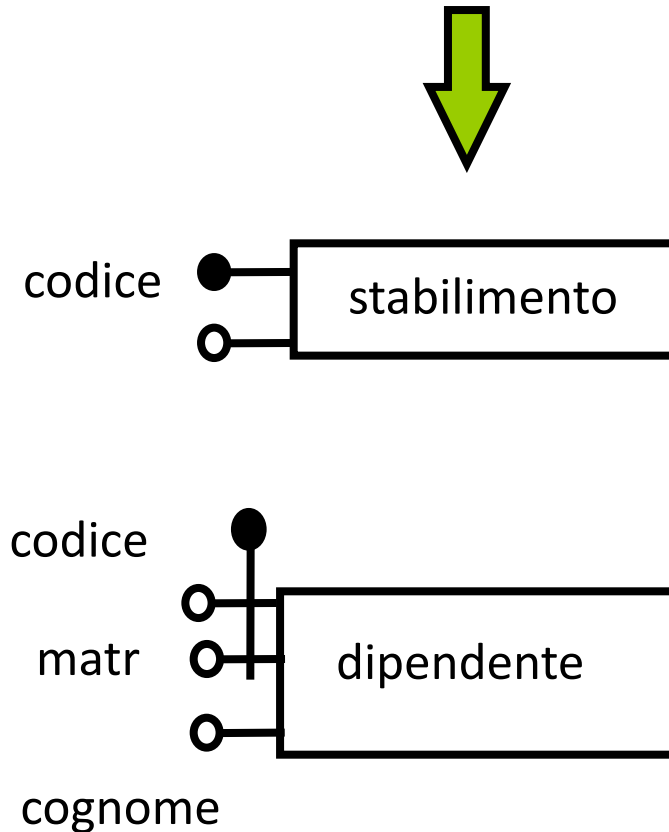
- una componente di identificazione esterna di una entità E2 da una entità E1 attraverso una associazione R comporta il **trasporto della chiave primaria** di E1 su E2





# identificatori esterni

- in questo modo l'associazione è rappresentata attraverso la chiave, e può essere eliminata
- la chiave trasportata è chiave esterna
- in presenza di più identificazioni in cascata, è necessario iniziare la propagazione dall'entità che non ha identificazioni esterne



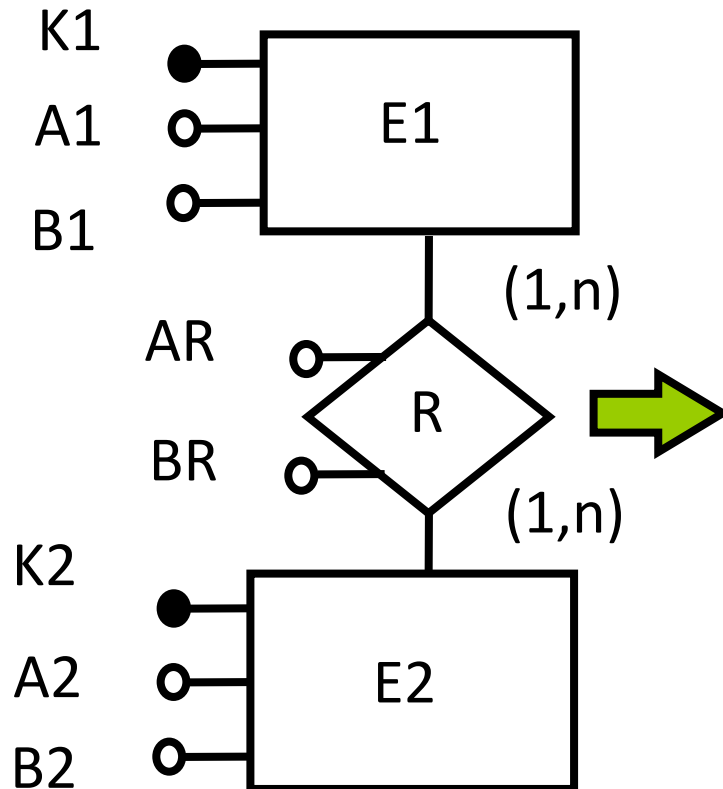
# Traduzione standard

## Entità ed Associazioni molti a molti

- ogni entità è tradotta con una relazione con gli stessi attributi
  - la chiave è la chiave (o identificatore) dell'entità stessa
- ogni associazione è tradotta con una relazione con gli stessi attributi, cui si aggiungono gli identificatori di tutte le entità che essa collega (già visto)
  - la chiave è composta dalle chiavi delle entità collegate

# Traduzione standard

## Entità ed Associazioni molti a molti

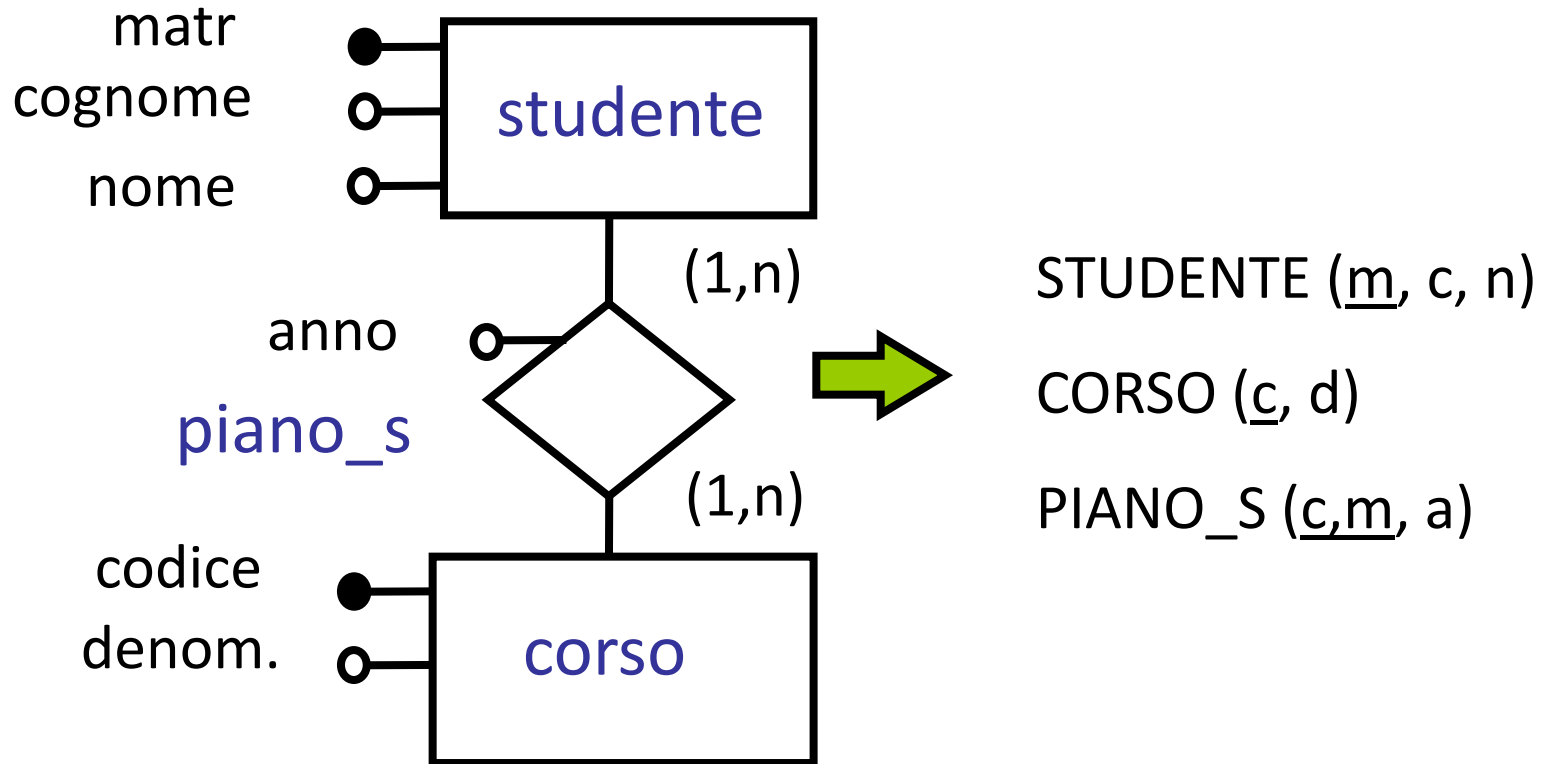


E1 (K1, A1, B1,...)

E2 (K2, A2, B2,...)

R (K1, K2, AR, BR,...)

# traduzione standard: es.



# traduzione standard: es.

```
CREATE TABLE STUDENTE (MATR... NOT NULL,  
..., NOME... , PRIMARY KEY (MATR));
```

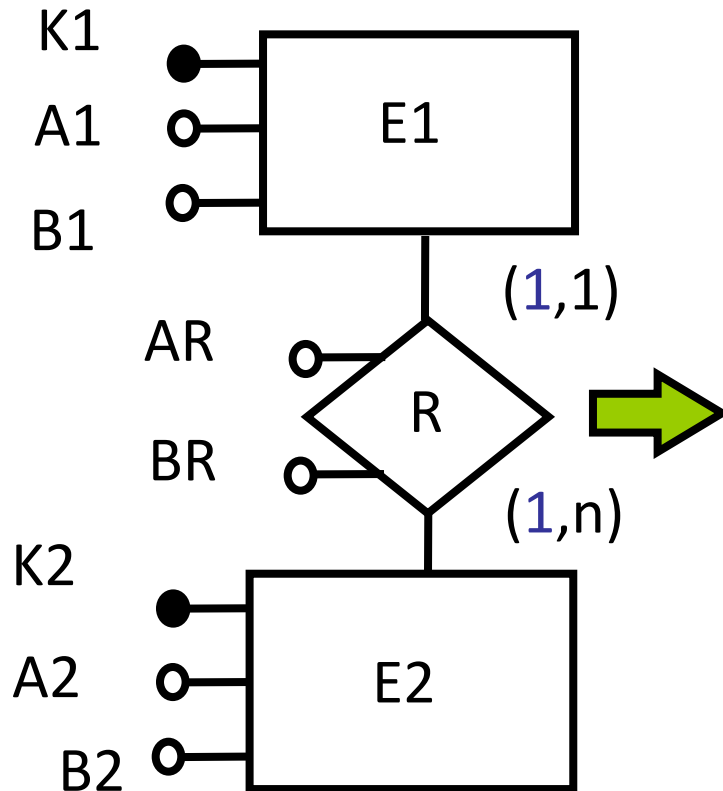
```
CREATE TABLE CORSO (CODICE... NOT NULL,  
DENOM ... , PRIMARY KEY (CODICE));
```

```
CREATE TABLE PIANO_ST (MATR... NOT NULL, CODICE... NOT NULL, ANNO...  
PRIMARY KEY (MATR, CODICE),  
FOREIGN KEY (MATR) REFERENCES STUDENTE  
FOREIGN KEY (CODICE) REFERENCES CORSO);
```

# altre traduzioni

- La traduzione standard è sempre possibile ed è l'unica possibilità per le associazioni  $N$  a  $M$
- Altre forme di traduzione delle associazioni sono possibili per altri casi di cardinalità (1 a 1, 1 a  $N$ )
- Le altre forme di traduzione fondono in una stessa relazione entità e associazioni

# Associazione binaria 1 a N



- traduzione standard:

E1 (K1, A1, B1)

E2 (K2, A2, B2)

R (K1, K2, AR, BR)

# associazione binaria 1 a N

- Se E1 partecipa con cardinalità (1,1) può essere fusa con l'associazione, ottenendo una soluzione a due relazioni:

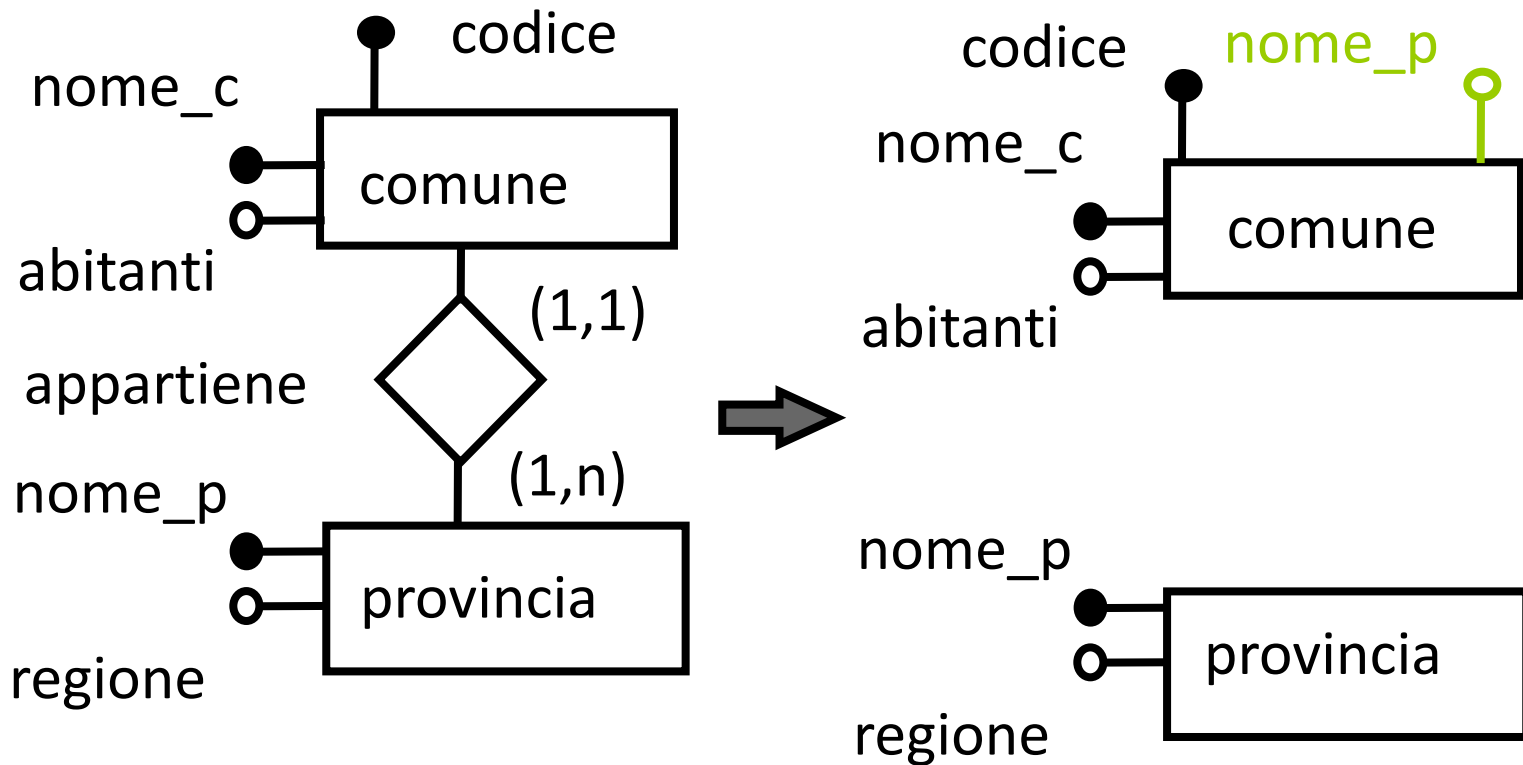
**E1(K1, A1, B1, K2, AR, BR)**

**E2(K2, A2, B2)**

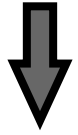
- Se E1 partecipa con cardinalità (0,1) la soluzione a due relazioni ha valori nulli in K2, AR, BR per le istanze di E1 che non partecipano all'associazione



# ass. binaria 1 a N es.



(senza attributi sull'associazione)

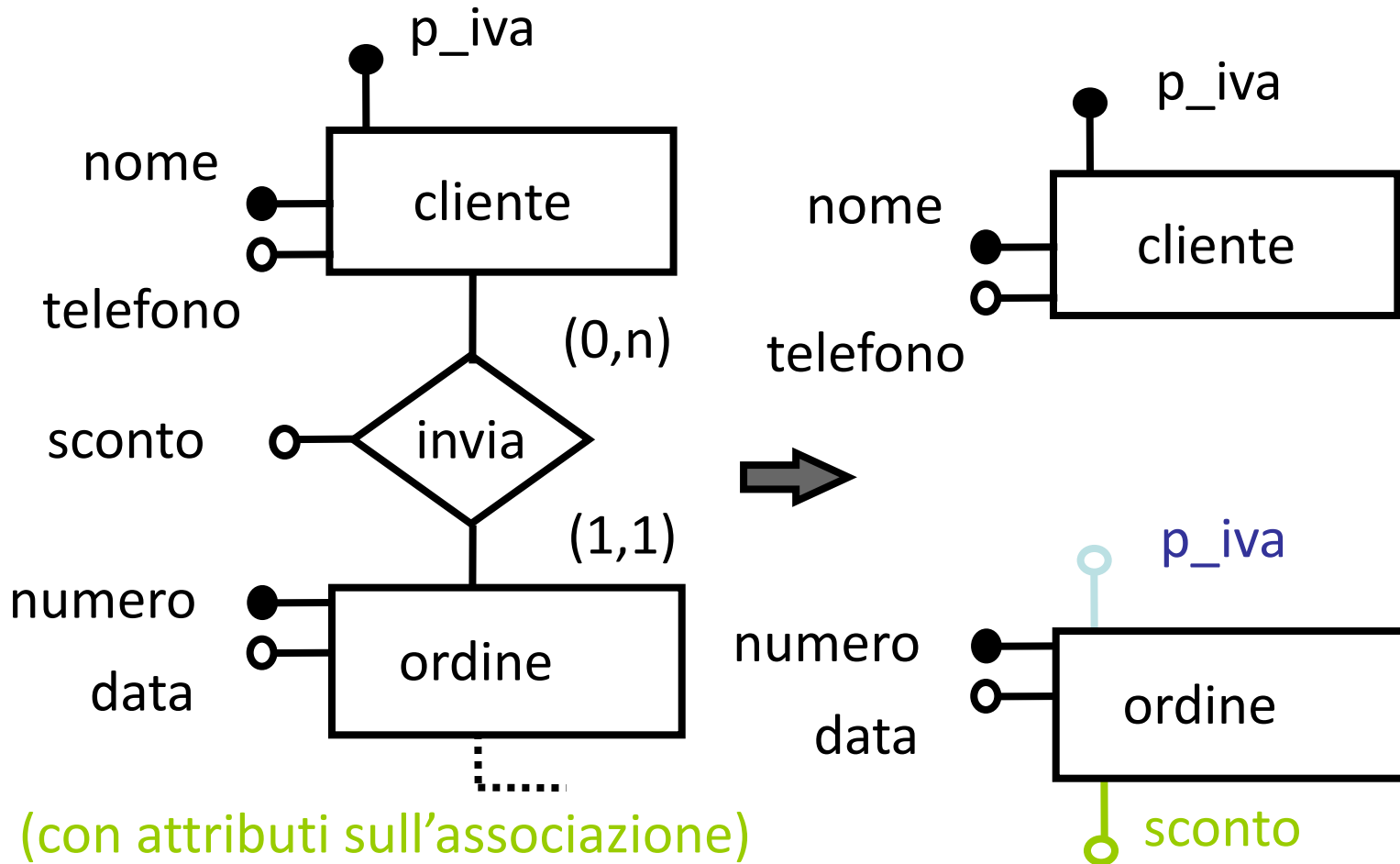


ass. binaria 1 a N es.

```
CREATE TABLE PROVINCIA  
(NOME_P ... NOT NULL,  
REGIONE ... PRIMARY KEY (NOME_P));
```

```
CREATE TABLE COMUNE  
(CODICE ... NOT NULL, NOME_C ...  
ABITANTI ..., NOME_P ... NOT NULL  
PRIMARY KEY (CODICE)  
FOREIGN KEY NOME_P  
REFERENCES PROVINCIA);
```

ass. binaria 1 a N es.



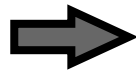
# ass. binaria 1 a N es.

traduzione con due relazioni:

```
CREATE TABLE CLIENTE (P_IVA..... NOT NULL, NOME  
...,TELEFONO ..., PRIMARY KEY (P_IVA));
```

```
CREATE TABLE ORDINE (NUMERO ... NOT NULL,  
    DATA ... P_IVA ... NOT NULL, SCONTO ..., PRIMARY KEY  
(NUMERO)  
FOREIGN KEY P_IVA REFERENCES CLIENTE);
```

con tre relazioni:



# ass. binaria 1 a N es.

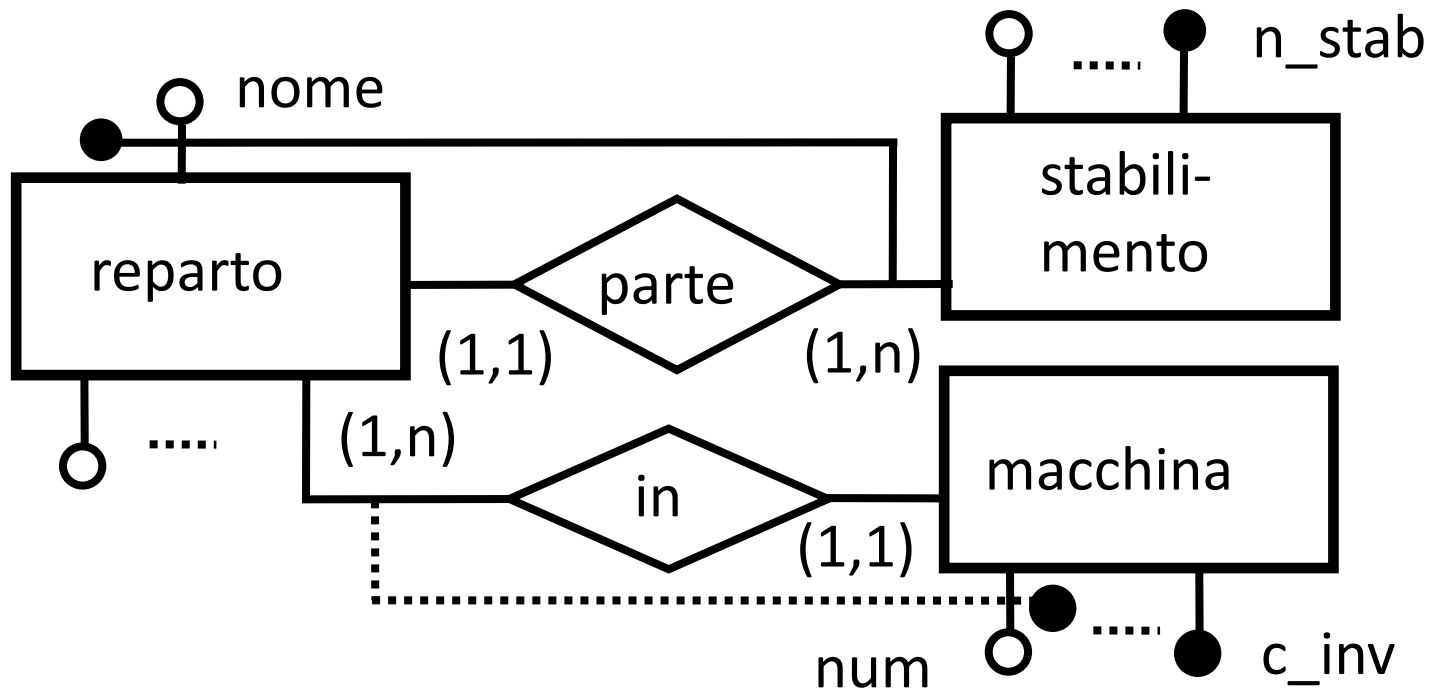
```
CREATE TABLE CLIENTE (P_IVA..... NOT NULL,  
    NOME ...,TELEFONO ..., PRIMARY KEY (P_IVA));
```

```
CREATE TABLE ORDINE (NUMERO ... NOT NULL,  
    DATA ... PRIMARY KEY (NUMERO));
```

```
CREATE TABLE SCRIVE  
(P_IVA ... NOT NULL, NUMERO ... NOT NULL,  
    SCONTO ..., PRIMARY KEY (NUMERO)  
    FOREIGN KEY P_IVA REFERENCES CLIENTE  
    FOREIGN KEY NUMERO REFERENCES  
    ORDINE);
```

# ass. binaria 1 a N es.

Con identificazione esterna

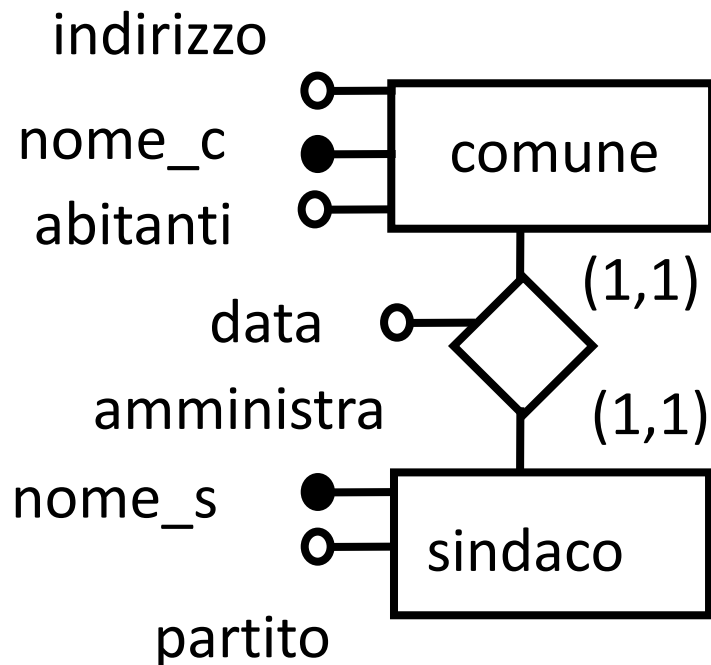


ass. binaria 1 a N es.

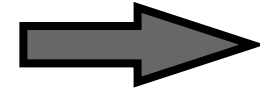
STABILIMENTO (N\_STAB .....);  
REPARTO (NOME, N\_STAB, .....);  
MACCHINA (NUM, NOME, N\_STAB);

# Associazione binaria 1 a 1

- traduzione con una relazione:



E12 (K1, A1, B1,  
K2, A2, B2,  
AR, BR)



Da escludere, lo schema che stiamo traducendo e' ristrutturato!  
se le cardinalità minime sono entrambe 1 la  
chiave può essere indifferentemente K1 o K2  
si sceglierà quella più significativa



# associazione binaria 1 a 1

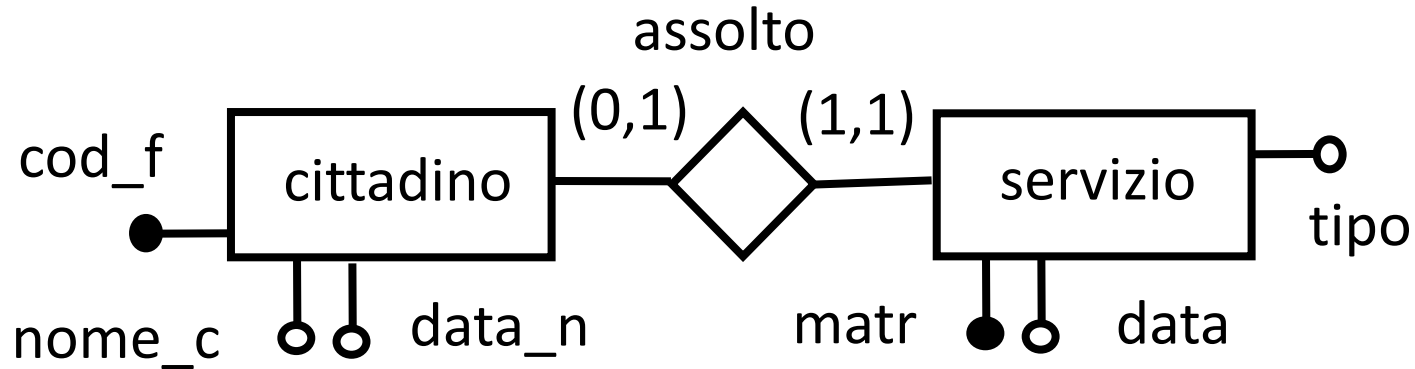
```
CREATE TABLE AMMINISTRAZIONE  
(NOME_C ... NOT NULL,  ABITANTI ...,  
  NOME_S ... NOT NULL UNIQUE,  
  INDIRIZZO ...,  DATA  
  PRIMARY KEY (NOME_C));
```

se le cardinalità minime sono entrambe 1 la  
chiave può essere indifferentemente K1 o K2  
si sceglierà quella più significativa

# associazione binaria 1 a 1

- se la cardinalità di E2 è 0,1 e quella di E1 è 1,1 allora la chiave sarà K2 ; E2 è l'entità con maggior numero di istanze alcune della quali non si associano, ci saranno quindi valori nulli in corrispondenza di K1, K1 in questo caso non potrebbe essere scelta

# associazione binaria 1 a 1



CITTADINO (**COD\_F**, NOME\_C, INDIRIZZO, DATA\_N, MATR, DATA, TIPO);

# associazione binaria 1 a 1

- Traduzione con due relazioni
  - l'associazione può essere **compattata** con l'entità che partecipa **obbligatoriamente** (una delle due se la partecipazione è obbligatoria per entrambe) la discussione sulla chiave è analoga al caso di traduzione con una relazione

E1 (K1, A1, B1,...)

E2 (K2, A2, B2,... K1, AR, BR)

# associazione binaria 1 a 1

- se la cardinalità è 0,1 da entrambe le parti allora le relazioni possono essere tre: la chiave della relazione che traduce l'associazione può essere indifferentemente K1 o K2, non ci sono problemi di valori nulli

E1 (K1, A1, B1,...)

E2 (K2, A2, B2,...)

R (K1, K2, AR, BR,...)

# Esempi di software

- DbDesigner presente all'interno di MySQL workbench.