Progetto Logico da Schemi E/R

- **Lo schema E/R descrive un dominio applicativo ad un livello astratto**
- **Lo schema E/R è molto utile per:**
  - **fornire una descrizione sintetica e visiva**
  - **rappresentare buona parte della semantica dell'applicazione**
  - **scambiare informazioni sull'attività progettuale tra i membri del team di progetto e mantenere una documentazione**
- **Non esistono DBMS in grado di operare direttamente sugli oggetti di uno schema E/R:**
  è quindi necessario tradurli in altri modelli di dati
- **Questa traduzione può essere eseguita in modo semi-automatico**
- **Esistono scelte alternative per motivi di efficienza**

Scelte alternative

- **Il progetto logico presenta in generale una soluzione standard determinabile *meccanicamente***
- **A seconda dei casi sono spesso disponibili anche soluzioni alternative**
- **Le soluzioni alternative possono rivelarsi più o meno convenienti, a seconda delle operazioni che saranno effettuate sui dati**
- **Per una scelta corretta sarebbero necessarie:**
  - **precise previsioni sulla natura e la frequenza delle operazioni**
  - **valutazioni quantitative sui volumi degli accessi alle varie informazioni**
  - **valutazioni quantitative sui volumi di dati (cardinalità, percentuali di copertura di gerarchie, percentuali di valori nulli)**

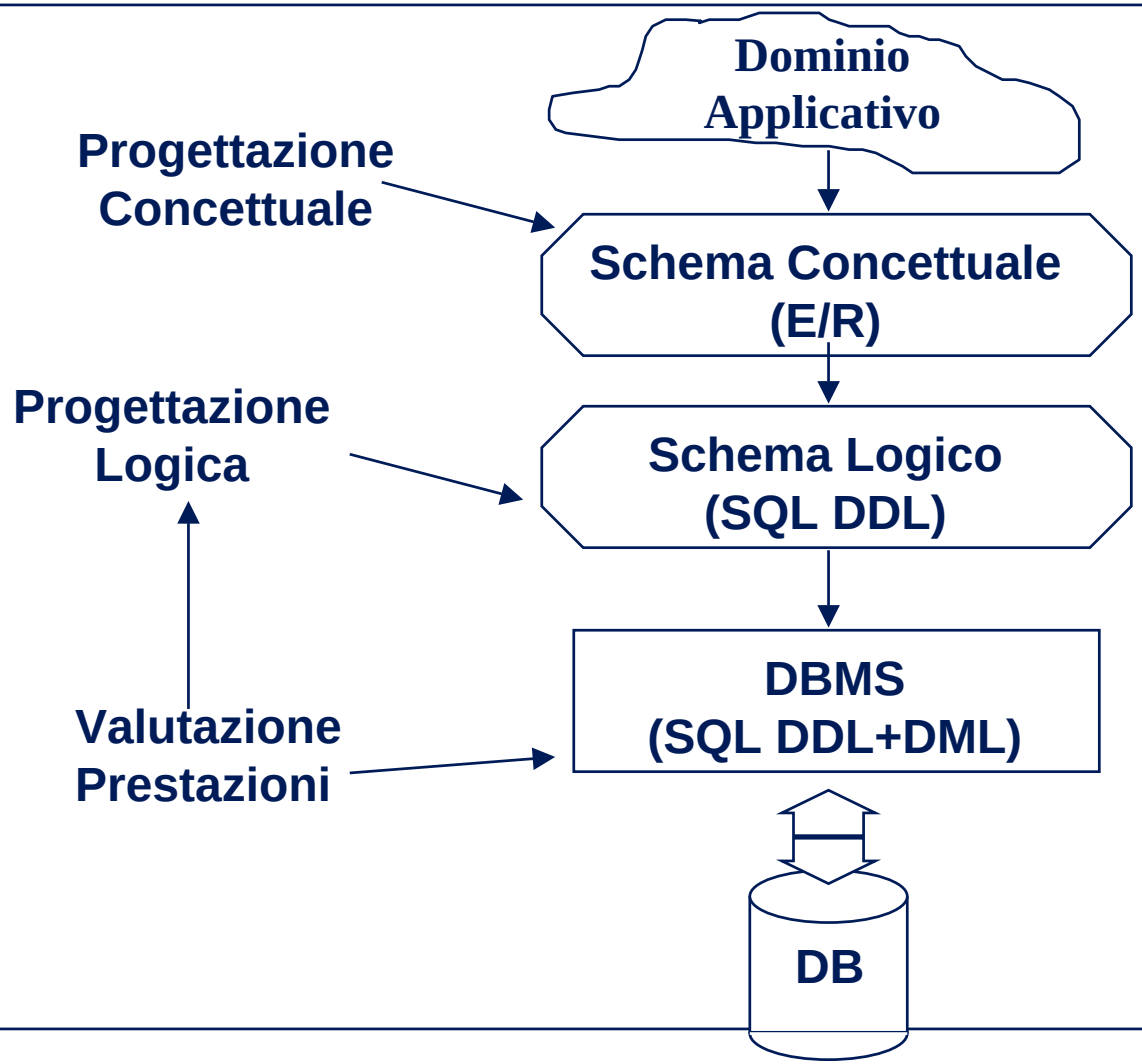
Scelte alternative (ii)

- **Si possono individuare alcune regole intuitive:**
  - **le proprietà logiche sono comunque primarie rispetto ai motivi di efficienza**
  - **tenere sulla stessa entità informazioni che verranno di frequente consultate insieme**
  - **tenere su entità separate informazioni che verranno consultate separatamente**
  - **limitare l'incidenza di valori nulli per attributi opzionali**

Progetto Logico da Schemi E/R (ii)

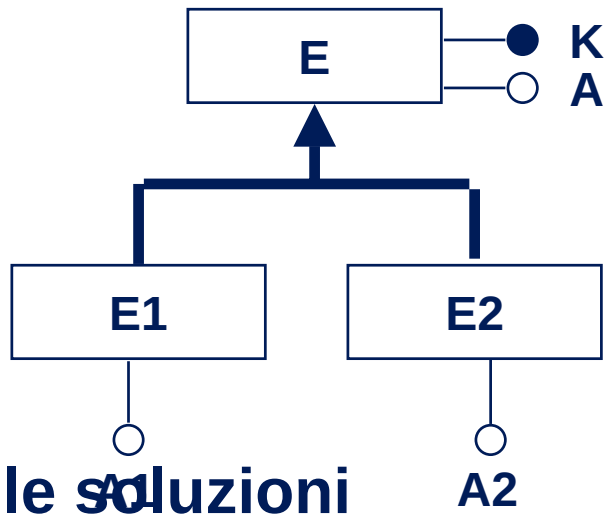
- **Fasi**
  - **trasformazione dello schema E/R con selezione di chiavi primarie, eliminazione di gerarchie e di identificazioni esterne**
  - **normalizzazione degli attributi composti o multipli**
  - **traduzione di entità e associazioni in schemi di relazioni**
- **Dopo le prime due fasi, lo schema E/R è costituito soltanto da entità, associazioni e attributi semplici, e la terza fase non è complessa**
- **Ogni trasformazione *impoverisce* lo schema; la semantica persa deve restare sotto forma di *vincoli di integrità* che governeranno l'utilizzo delle relazioni**

Processo di design



Eliminazione delle gerarchie

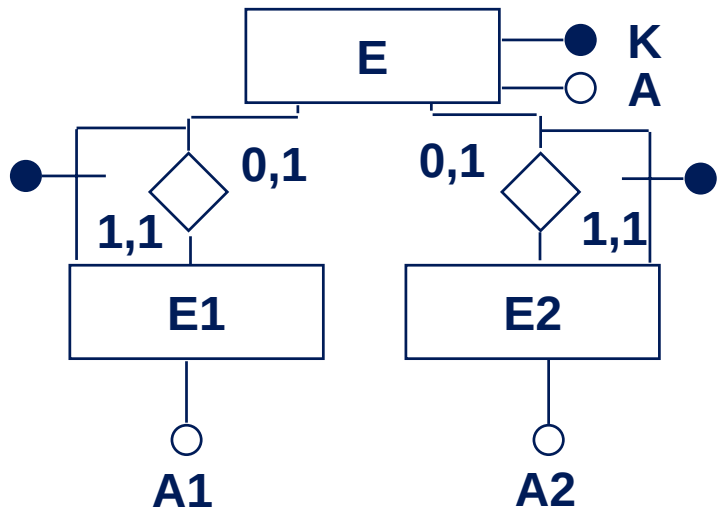
- Il modello relazionale non rappresenta le gerarchie
- Le gerarchie sono sostituite da entità e associazioni
- 3 possibilità: *mantenimento delle entità con associazioni, collasso verso l'alto*, *collasso verso il basso*



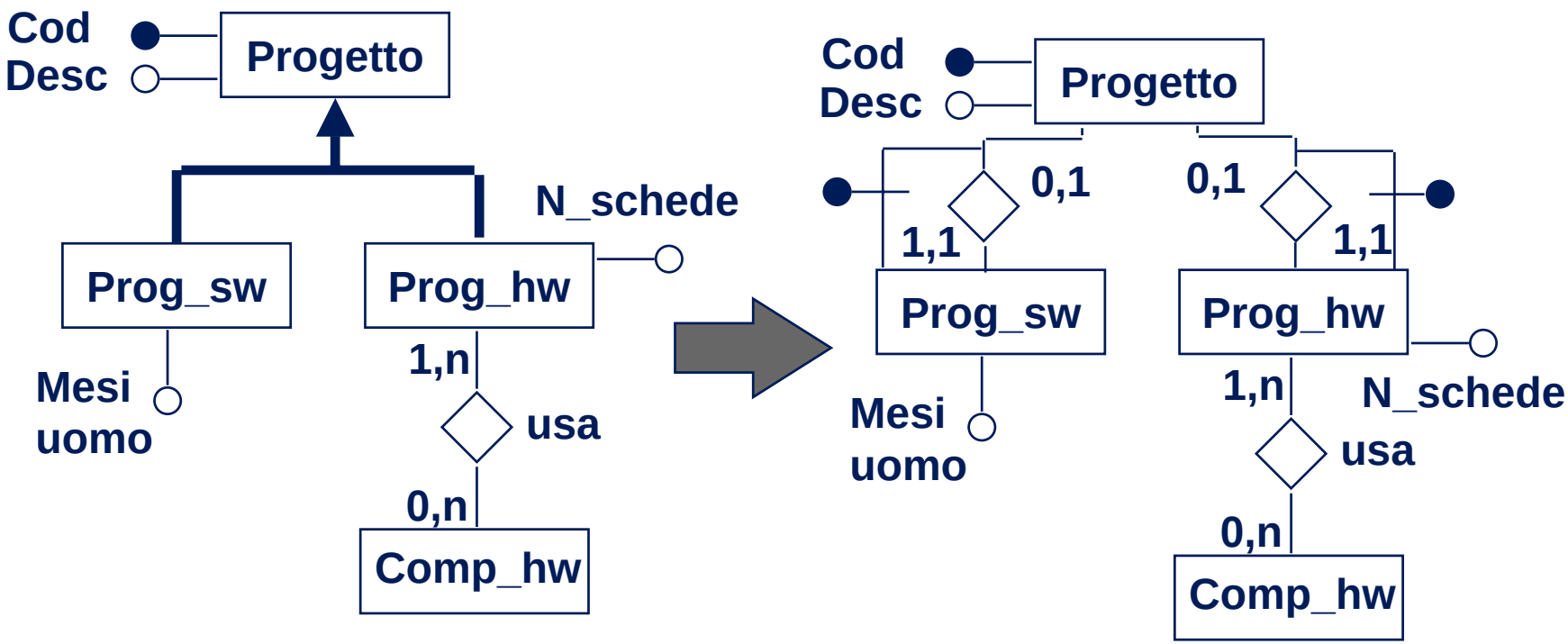
- l'applicabilità e la convenienza delle soluzioni dipendono dalle proprietà di copertura e dalle operazioni previste
  - esempio di operazione: fornire i valori di A e A1 per tutti gli E1

Mantenimento delle entità con associazioni

- 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 delle entità con associazioni - esempio



Eliminazione delle gerarchie - alternative

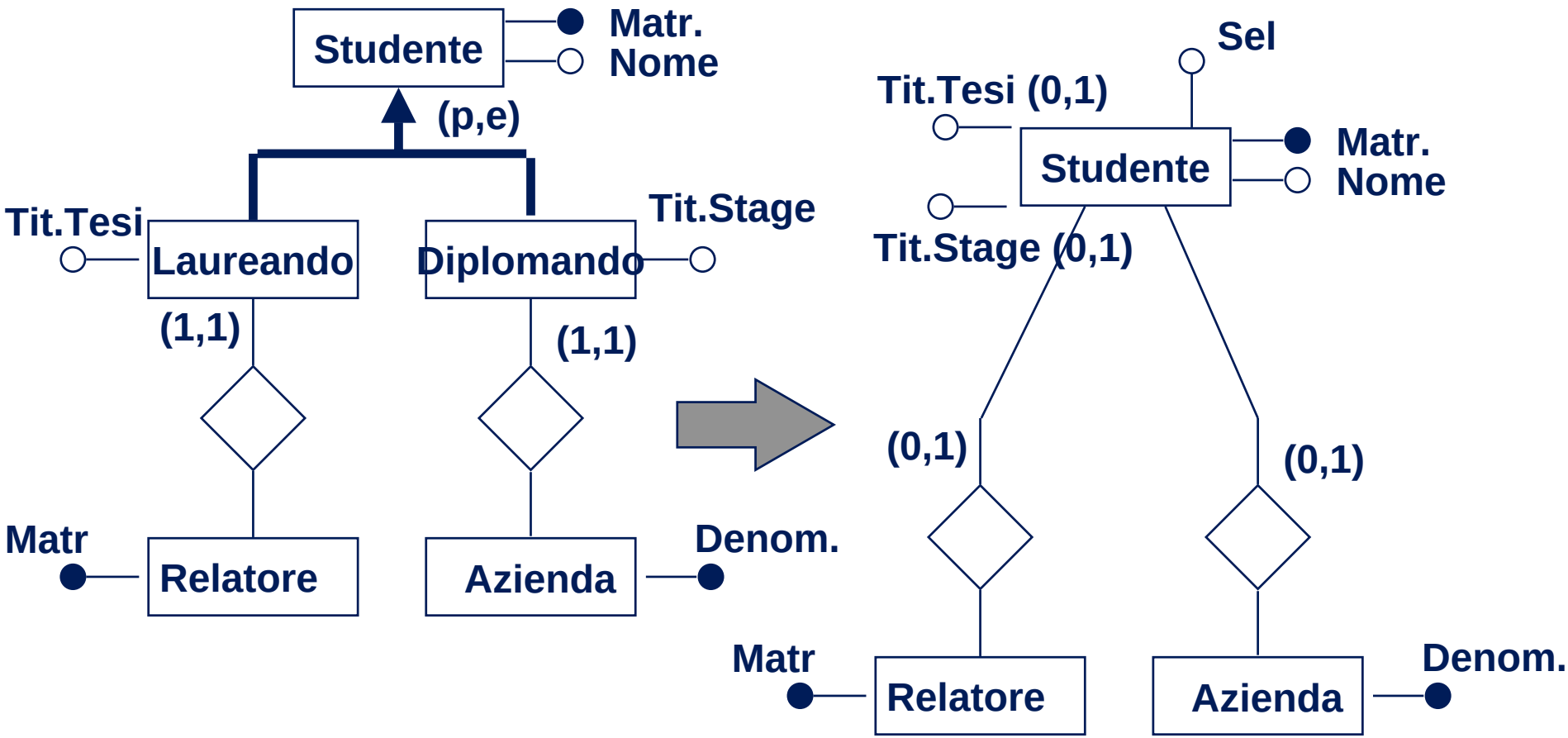
- Il *collasso verso l'alto* riunisce tutte le entità figlie nell'entità padre
- Questa soluzione favorisce operazioni che consultano *insieme* gli attributi dell'entità padre e quelli di una entità figlia:
  - in questo caso si accede a una sola entità, anziché a due attraverso una associazione
- Gli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

Collasso verso l'alto

- **"Sel" è un attributo *selettore* che specifica se una singola e istanza di E appartiene a una delle N sottoentità**

- **copertura**
  - ***totale esclusiva:*** **Sel ha N valori, quante sono le sottoentità**
  - ***parziale esclusiva:*** **Sel ha N+1 valori; il valore in più serve per le istanze che non appartengono a nessuna sottoentità**
  - ***sovrapposta:*** **occorrono tanti selettori quante sono le sottoentità, ciascuno a valore booleano Sel_i, che è vero per ogni istanza di E che appartiene a E_i; se la copertura è *parziale* i selettori possono essere tutti falsi, oppure si può aggiungere un selettore**
- **le eventuali associazioni connesse alle sottoentità si trasportano su E, le eventuali cardinalità minime diventano 0**

Collasso verso l'alto - esempio



Dom(Sel)=(L,D,N)

Collasso verso l'alto - esempio (ii)

- studente(s1,123,rossi)
  studente(s2,218,bianchi)
  studente(s3,312,verdi)
- laureando(s1,sist_inf)
- diplomando(s2,XYZ_SPA)

- studente(s1,123,rossi,L,
  sist_inf,NULL)
  studente(s2,218,bianchi,D,
  NULL,XYZ_SPA)
  studente(s3,312,verdi,N,
  NULL,NULL)

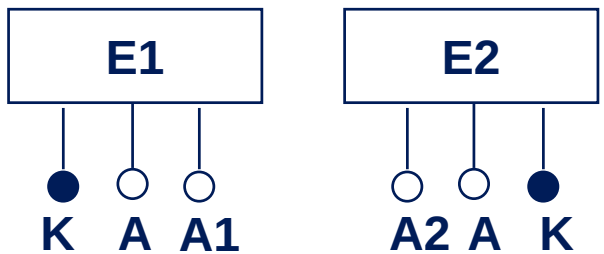
- esiste una precisa relazione tra il valore del selettore e i campi che possono avere valore diverso da NULL
- campi prima obbligatori ora ammettono NULL
- per una stima delle percentuali di NULL occorre conoscere le percentuali di laureandi e diplomandi

Eliminazione delle gerarchie - alternative

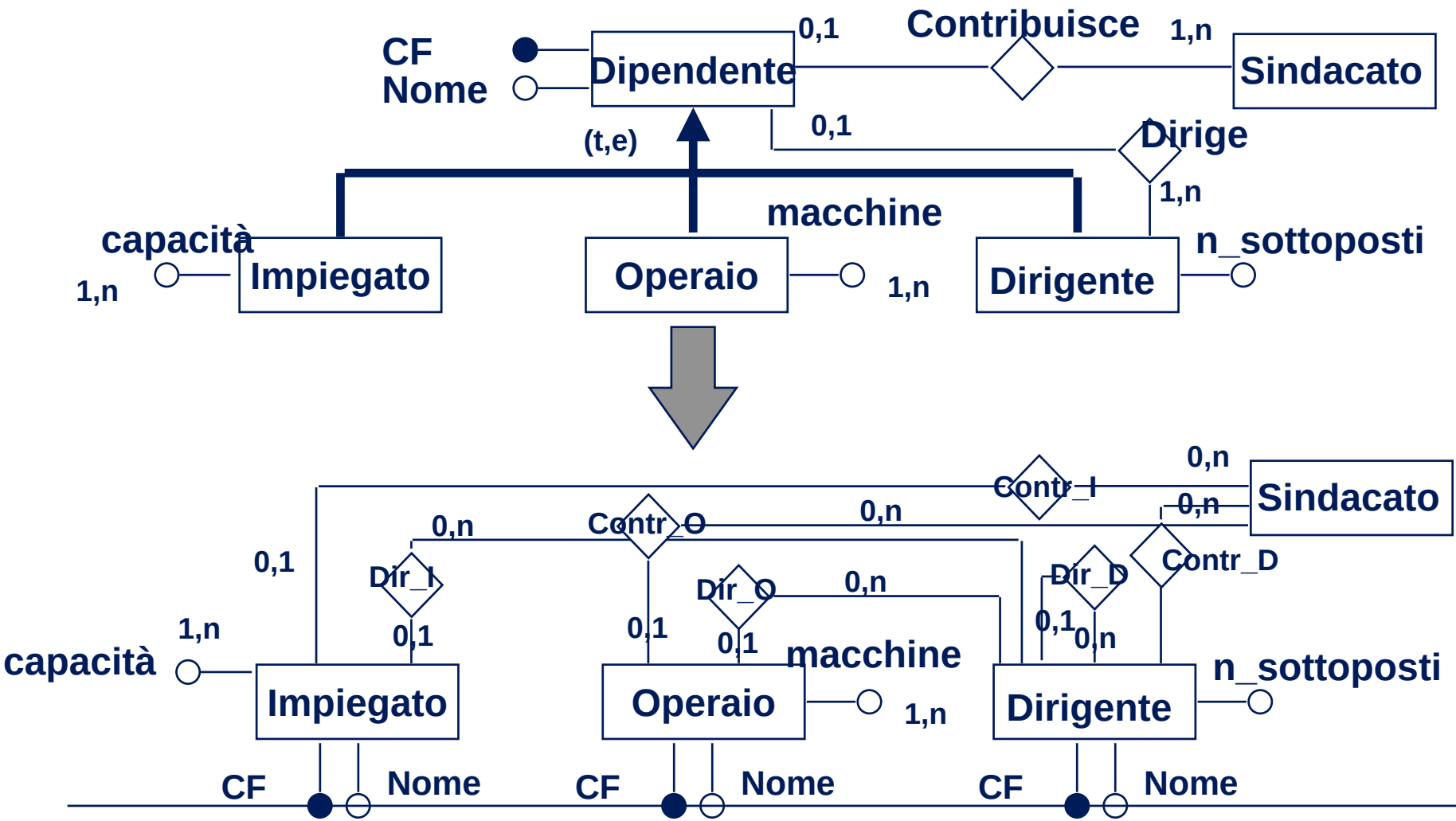
- Il *collasso verso il basso* elimina l'entità padre trasferendone attributi e associazioni su tutte le entità figlie
- Una associazione del padre si *moltiplica,* tante volte quante sono le entità figlie
- Soluzione interessante in presenza di molti attributi di specializzazione
- Favorisce le operazioni in cui si accede separatamente a ciascuna delle entità figlie
  - esempio: fornire i valori di A per tutti gli E1
- Esistono limiti di applicabilità

Collasso verso il basso

- **Se la copertura non è completa *non si può fare***
  - **non si saprebbe dove mettere gli E che non sono né E1, né E2**
- **Se la copertura non è esclusiva introduce ridondanza**
  - **una certa istanza può essere sia in E1 che in E2, rappresentando due volte gli attributi che provengono da E**



Collasso verso il basso - esempio



Scelta della chiave primaria

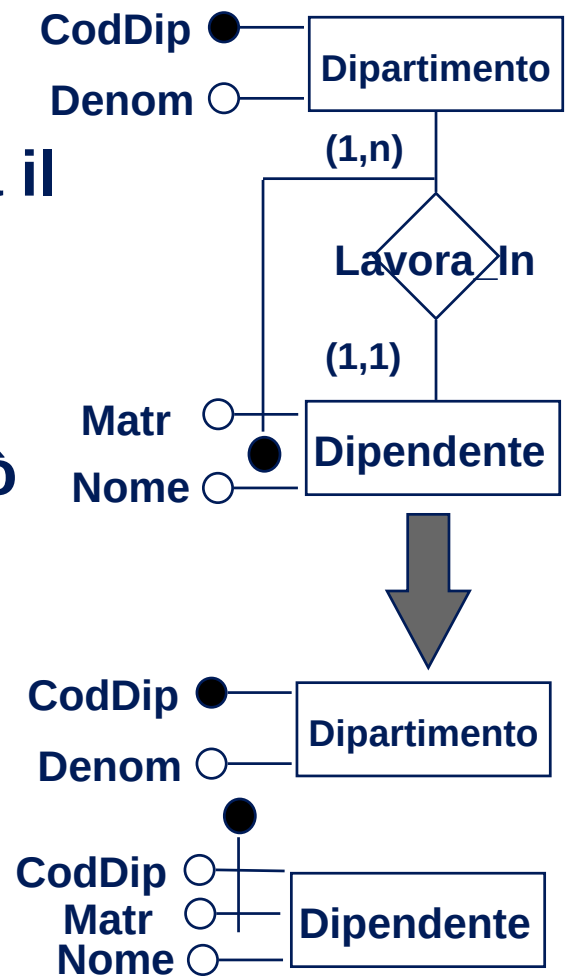
- **Per motivi di implementazione, è necessario che tra i diversi identificatori di una entità venga designata una *chiave primaria*, che verrà trattata in modo particolare dal sistema**

- **Criteri euristici:**
  - **primario: scegliere la chiave che è usata più frequentemente per accedere all'entità**
  - **secondario: si preferiscono chiavi semplici a chiavi composte, interne anziché esterne**

- **Per tutti gli identificatori occorrerà, in fase di implementazione sul DBMS, prevedere strumenti per garantire l'unicità dei valori**

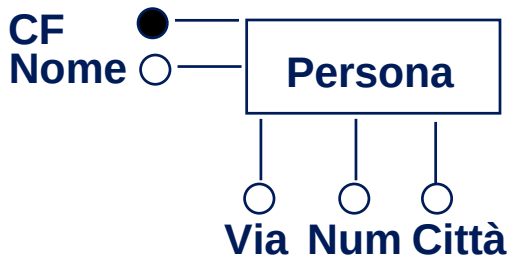
Eliminazione degli identificatori esterni

- **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**
- **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**



Normalizzazione degli attributi composti

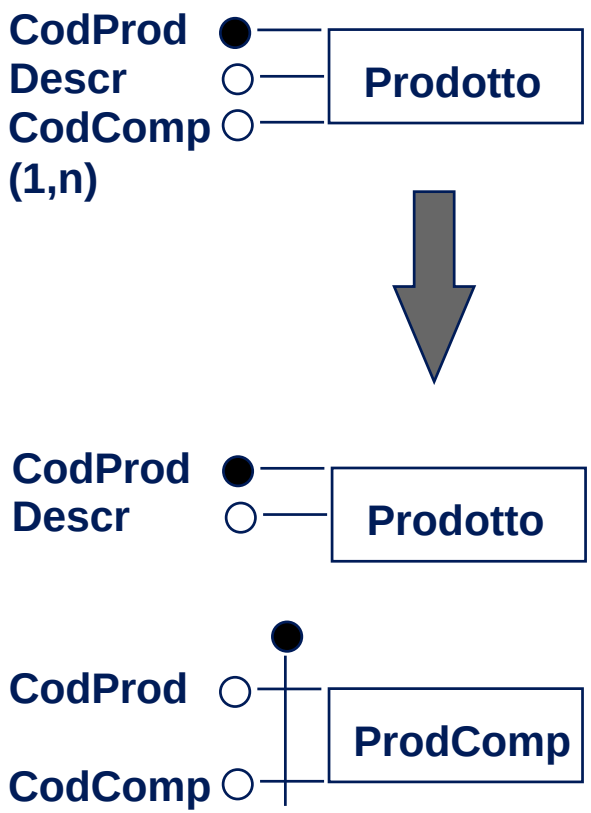
- **Due possibili soluzioni**
  - **eliminare l'attributo composto e considerare i suoi componenti come attributi semplici**
    - » **in questo modo si perde la visione unitaria ma si mantiene l'articolazione dei componenti**
  - **eliminare i componenti e considerare l'attributo come semplice**
    - » **in questo modo lo schema risulta semplificato, perdendo parte dei dettagli**

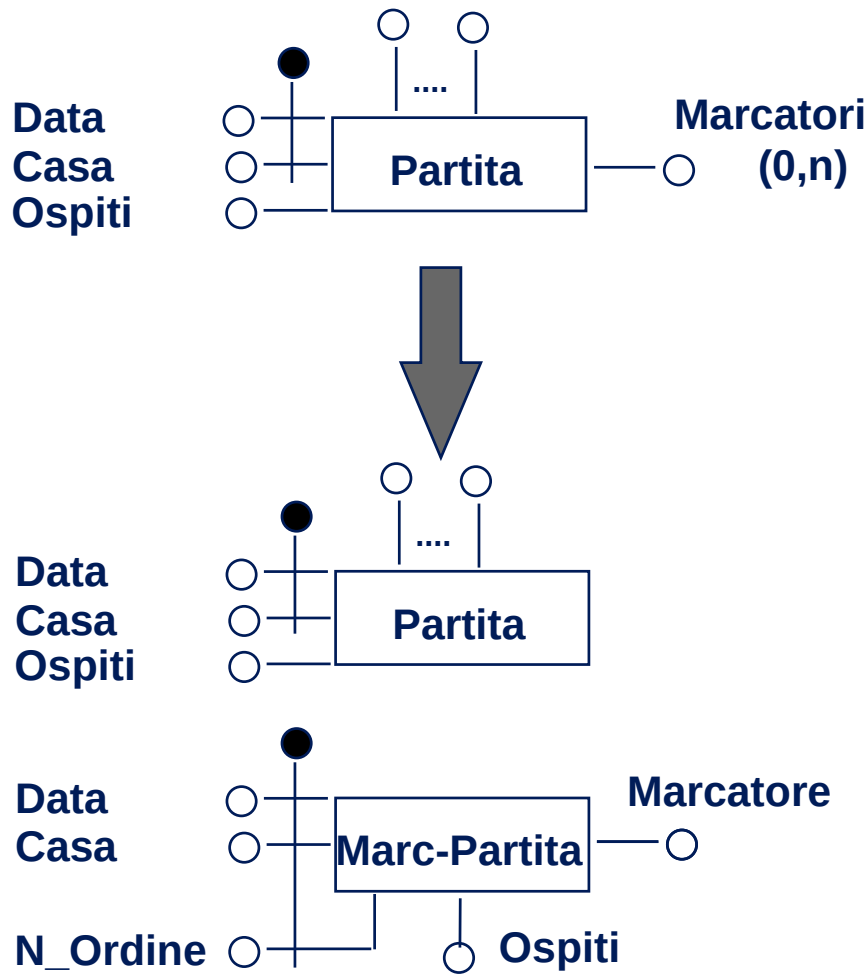


Normalizzazione degli attributi ripetuti

- **La prima forma normale *impone* che, se una entità E ha un attributo A ripetuto, si crei una nuova entità che contiene l'attributo ed è collegata a E**
  - **Caso a) - un valore può comparire una volta sola nella ripetizione**
    - **la nuova entità EA ha l'identificatore composto dall'identificatore di E più l'attributo A**
  - **Caso b) - un valore può comparire più volte nella ripetizione**
    - **la nuova entità EA ha l'identificatore composto dall'identificatore di E più un valore identificante sintetico (ad esempio, un numero d'ordine)**
- **La chiave di E è chiave esterna per EA**
- **Caso particolare: l'attributo può essere ripetuto *al più K volte***
  - **in questo caso l'attributo si può sostituire con K attributi dell'entità, il cui valore sarà non nullo per i primi H quando la ripetizione è H<K**

Normalizzazione degli attributi ripetuti - esempio



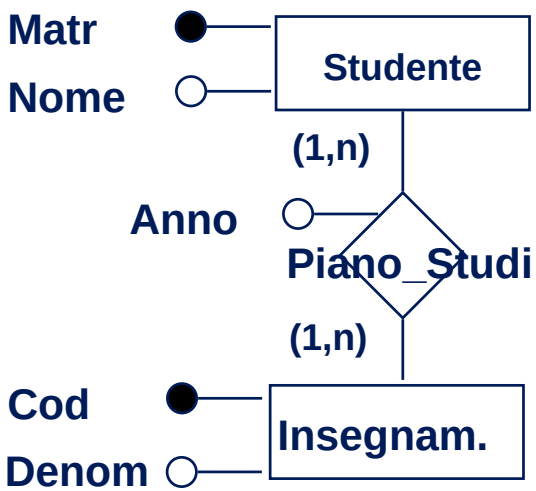


Traduzione standard

- **Ogni entità è tradotta con una relazione con gli stessi attributi**
  - **l'identificatore è quello 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**
  - **gli identificatori delle entità collegate costituiscono una *superchiave***
  - **la chiave si individua sulla base delle cardinalità delle entità nell'associazione**

```
E1(K1,A1,B1,...)
E2(K2,A2,B2,...)
R(K1,K2,AR,BR,...)
 FK:   (K1)->E1,  (k1 sulla chiave primaria di E1)
       (K2)->E2   (k2 sulla chiave primaria di E2)
```

Traduzione standard - esempio




```
Studente(Matr,Nome)
   PK: (Matr)
Insegnamento(Cod,Denom)
   PK: (Cod)
Piano_Studi(Matr,Cod,Anno)
   PK: (Matr, Cod)
   FK: (Matr) -> Studente
       (Cod) ->Insegnamento
```

Traduzione Standard (ii)

- **Nel caso uno a molti vi è una dipendenza funzionale da K1 a K2, quindi la chiave della relazione che traduce l'associazione è *ridotta***

  E1(<u>K1</u>,A1,B1,...)
  E2(<u>K2</u>,A2,B2,...)
  R(<u>K1</u>,K2,AR,BR,...)

- **Nel caso uno a uno**

  E1(<u>K1</u>,A1,B1,...)
  E2(<u>K2</u>,A2,B2,...)
  R(<u>K1</u>,K2,AR,BR,...)

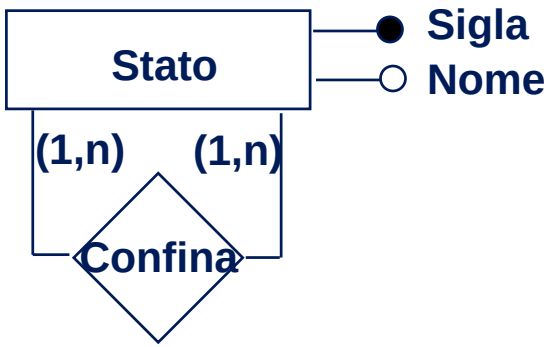
  – **la chiave della relazione che traduce l'associazione può essere indifferentemente K1 o K2**

Altre traduzioni

- **La traduzione standard *è sempre possibile* ed è l'*unica* possibilità per le associazioni molti a molti**
- **Altre forme di traduzione dell'associazione sono possibili *per casi particolari di cardinalità***
- **Le altre forme di traduzione:**
  - **tendono a fondere in una stessa relazione entità e associazioni**
  - **danno origine a un minore numero di relazioni e generano quindi uno schema più semplice**
  - **richiedono un minore numero di join per la *navigazione* attraverso una associazione, ovvero per conoscere le istanze di E1 connesse a E2 tramite R**
  - **penalizzano le operazioni che consultano soltanto gli attributi di una entità che è stata fusa**

Anello molti a molti

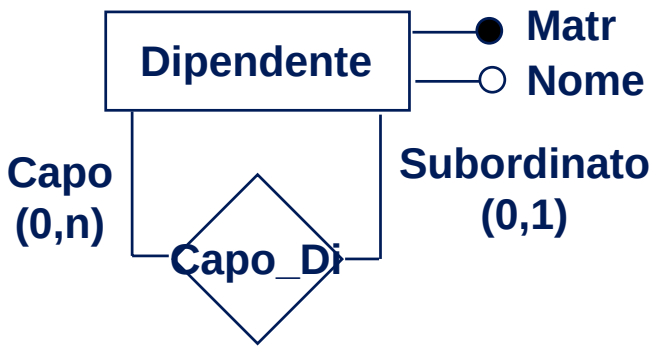
- **È tradotta con una relazione per l'entità e una per l'associazione; quest'ultima contiene due volte l'identificatore dell'entità**
- **Nella relazione dell'associazione è necessario modificare i nomi degli attributi, per non avere omonimia**




```
Stato(Sigla,Nome)
       PK: (Sigla)
Confina(Stato_a, Stato_b)
   PK: (Stato_a, Stato_b)
   FK: (Stato_a) -> Stato
       (Stato_b) -> Stato
```

Anello uno a molti

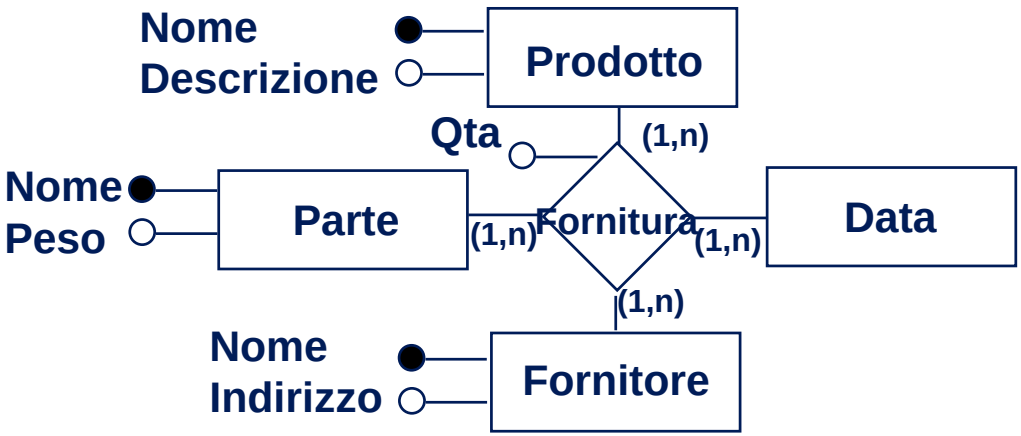
- **È traducibile con *una sola relazione* che contiene due volte l'attributo identificatore: una volta come riferimento all'istanza di entità connessa, e con nome diverso per specificare il ruolo**




```
Dipendente(Matr, Nome, Capo)
   PK: (Matr)
   FK: (Capo) -> Dipendente
```

Associazione n-aria

- **Segue la traduzione standard**

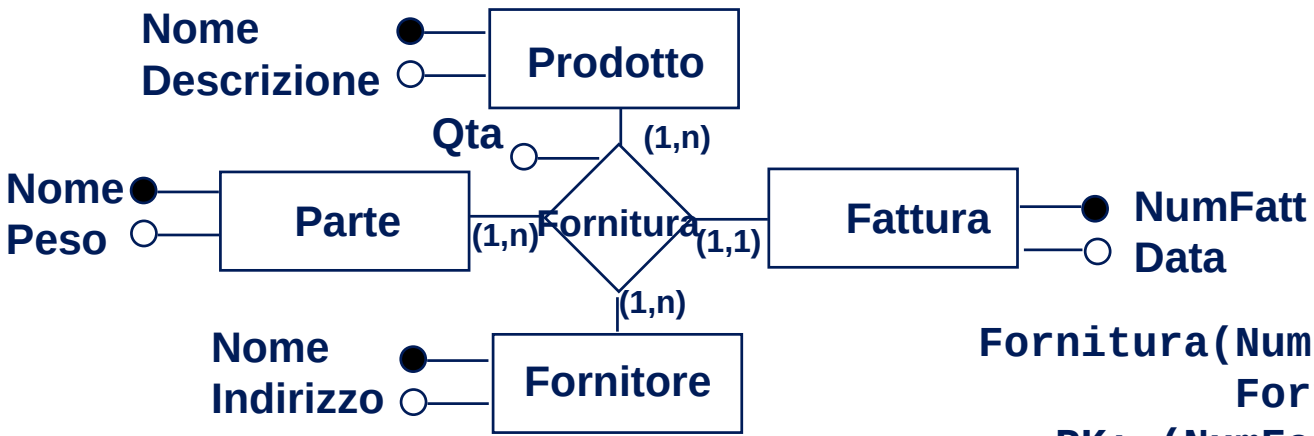



```
Prodotto(Nome, Descrizione)
   PK: (Nome)
Parte(Nome, Peso)   PK: (Nome)
Fornitore(Nome, Indirizzo)
   PK: (Nome)
Fornitura(Prodotto, Parte,
          Data, Fornitore, Qta)
   PK: (Prodotto, Parte,
        Fornitore, Data)
   FK: (Prodotto) -> Prodotto
       (Fornitore)-> Fornitore
       (Parte) -> Parte
```

NB: la data non è rappresentata come relazione a se stante, poiché in generale i sistemi sono in grado di fare una verifica automatica sulla sua validità

Associazione n-aria (ii)

- **Talvolta nella relazione che traduce l'associazione la chiave ottenuta con le chiavi di tutte le entità partecipanti è una superchiave**




```
Fornitura(NumFatt, Prodotto, Parte,
          Fornitore, Qta)
   PK: (NumFatt)
   FK: (NumFatt)  -> Fattura
       (Prodotto) -> Prodotto
       (Fornitore)-> Fornitore
       (Parte) -> Parte
```

Associazione binaria uno a molti

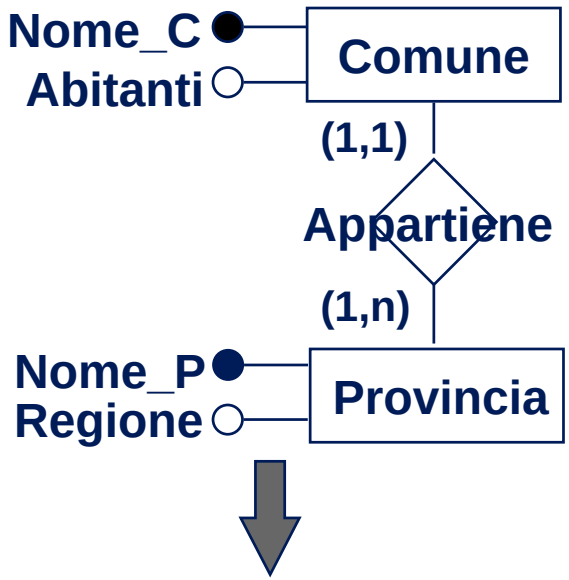
- **Se E1 ha cardinalità (1,1) può essere fusa con l'associazione, ottenendo una soluzione a due relazioni**

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

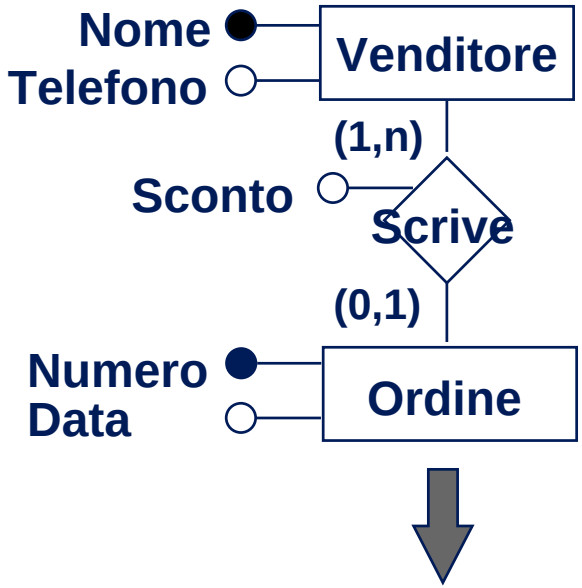
  E2(K2,A2,B2,...)

- **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**

Associazione binaria uno a molti - esempi



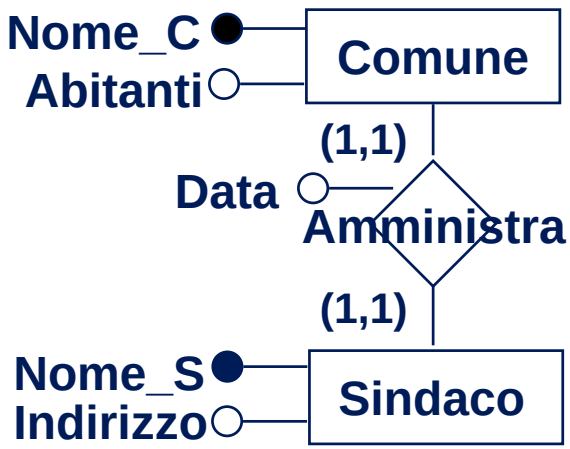

```
Provincia (Nome_P,Regione)
  PK: (Nome_P)
Comune(NOME_C,Abitanti,Nome_P)
   PK: (Nome_C)
   FK: Nome_P -> Provincia
```




```
Venditor(Nome,Telefono) PK: (Nome)
Ordine(Numero,Data)   PK: (Numero)
Scrive(Nome,Numero,Sconto)
   PK: (Numero)
   FK: (Nome) -> Venditor
       (Numero) -> Ordine
```

Associazione binaria uno a uno

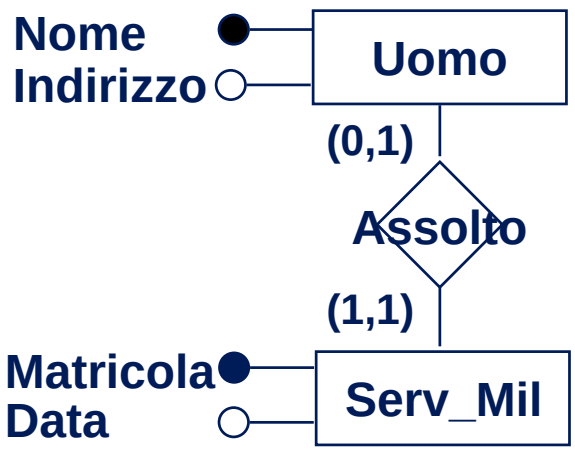
- **Traduzione con una relazione**
  E12(K1,A1,B1,...,K2,A2,B2,...,AR,BR,...)
  - **se le cardinalità minime sono entrambe 1 (associazione obbligatoria per entrambe le entità) la chiave può essere indifferentemente K1 o K2 (l'altra sarà chiave alternativa)**

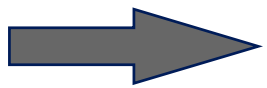



```
Amministrazione(Nome_C,Abitanti,
   Nome_S, Indirizzo,Data)
   PK: (Nome_C)
   Alternate Key: (Nome_S)
```

Associazione binaria uno a uno (cont.)

– se la cardinalità di E1 è (0,1) e quella di E2 è (1,1) la chiave *deve* essere K1 e saranno possibili valori nulli per gli attributi di E2 e di R
  - » *osservazione:* l'entità E1, con cardinalità (0,1) è quella con il maggior numero di istanze, alcune delle quali non hanno un corrispondente in E2



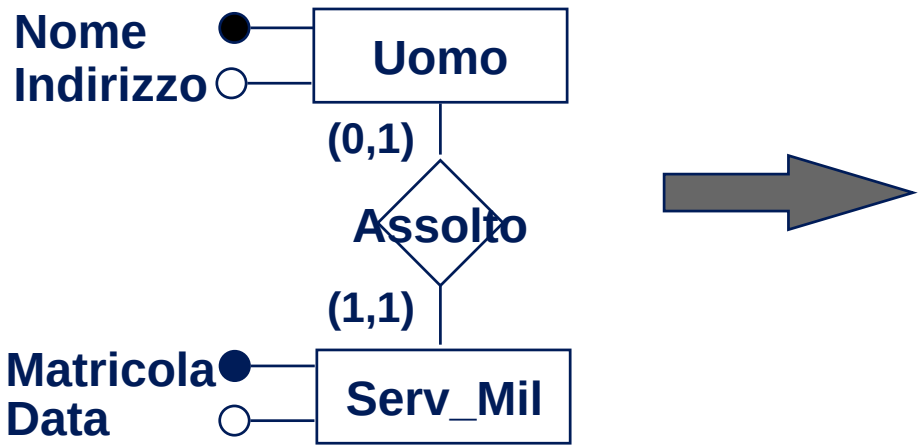


```
Uomo(Nome,Indirizzo,
     Matricola,Data)
PK: (Nome)
```

– se la cardinalità è (0,1) per entrambe, la traduzione con una sola relazione non è ammessa, per l'impossibilità di assegnare un identificatore

Associazione binaria uno a uno (cont.)

- **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**




```
Uomo(Nome,Indirizzo)
   PK: (Nome)
Serv_Mil(Nome,
      Matricola,Data)
   PK: Matricola
   FK: (Nome)->Uomo
```

Controllo vincoli di integrità

- **integrità di dominio**
  - in fase di creazione tabelle i domini sono molto ampi (interi, stringhe di lunghezza fissa o variabile, ...)
  - è possibile specificare ad esempio insiemi di stringhe
- **integrità di entità**
  - problemi legati alle chiavi e ai valori nulli
- **integrità referenziale**
  - problemi legati alle foreign key
- **integrità di utente**
  - legati alla singola tupla
  - legati a una relazione nel suo complesso
  - legati a più relazioni

Integrità di dominio

- si specificano i valori ammessi per un attributo
  - `CHECK (Sel IN 'L', 'D', 'N')`
  - ```
    CREATE DOMAIN Boolean AS CHARACTER(1)
      CHECK (VALUE IN 'T', 'F')
    ```
- i valori ammessi possono essere ricavati dinamicamente tramite una query
  - ```
    CREATE DOMAIN ColoriPresenti
      AS CHARACTER(20)
      CHECK (VALUE IN
            (SELECT Colore FROM Parti))
    ```
  - supponendo che esista una tabella Parti con l'attributo Colore
  - in questo modo le modifiche nella tabella Parti influenzano il dominio

Integrità di entità

- **chiave primaria**
  - `PRIMARY KEY`
  - unica e a valori nulli
- **chiavi alternative**
  - `NOT NULL`
  - `UNIQUE`

Integrità referenziale

- **`FOREIGN KEY (Fk1,Fk2)`**
  `REFERENCES Tabella`
  - **ci si riferisce alla chiave primaria della tabella referenziata**
  - **è più forte del vincolo espresso con CHECK, che non richiede chiave primaria**
  - **le modifiche in violazione vengono *bloccate***
  - **possono essere definite *azioni di mantenimento* che in caso di modifica sulla tabella referenziata *propagano* la modifica sulle righe opportune della tabella che effettua il riferimento**

Integrità di utente

- **a livello di tupla**
  - `CHECK (NOT (Sel1 = 'F' AND Sel2='F' ... AND Seln='F'))`
- **tra tabelle**
  - **CHECK ((SELECT COUNT(*)**
    FROM E1,E2
    WHERE E1.K = E2.K
) = 0
)