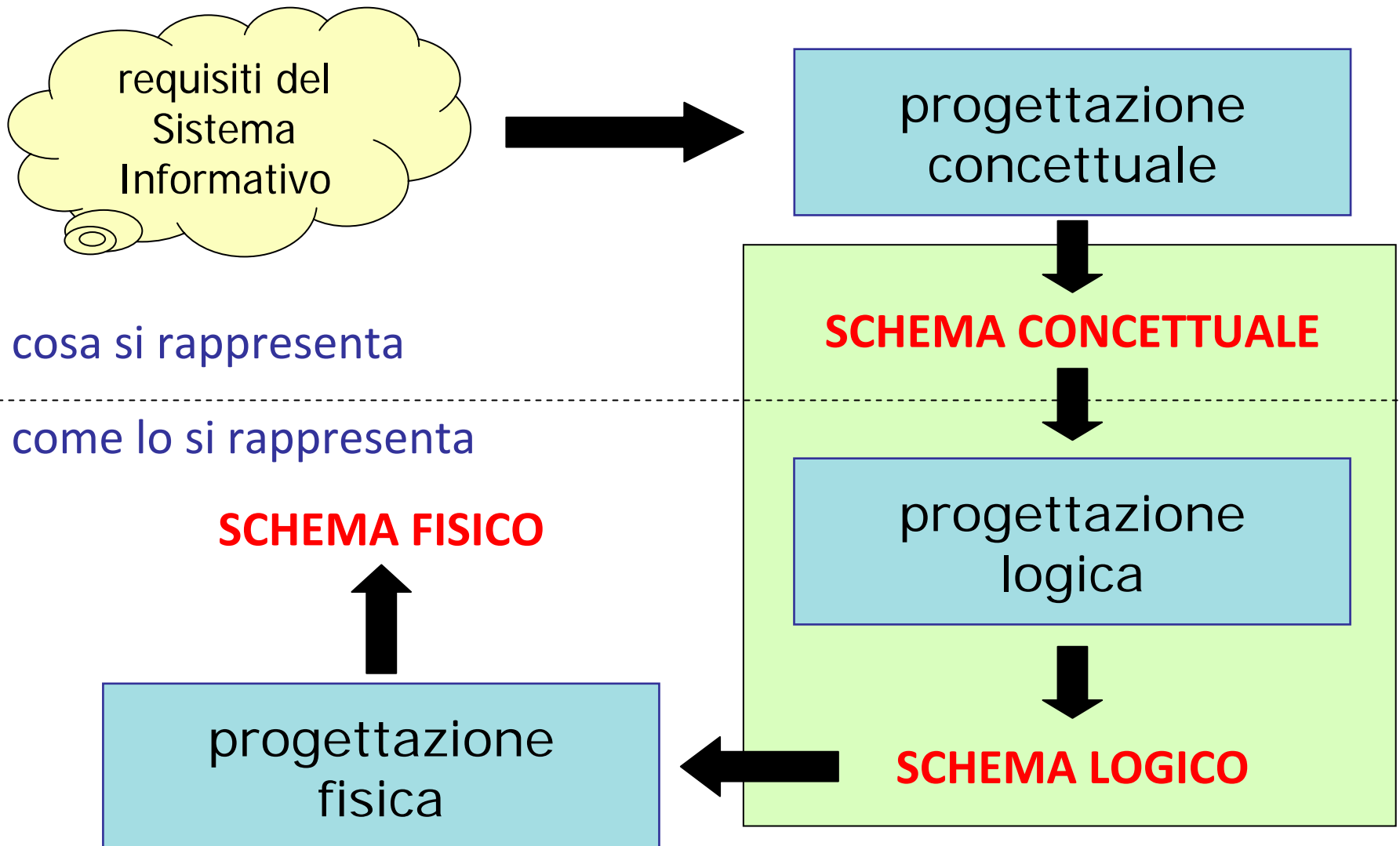


Progettazione logica: regole di traduzione

Sistemi Informativi T

Versione elettronica: [08.1.progLogica.traduzione.pdf](#)

Il secondo passo...



Progettazione logica

- Obiettivo della fase di progettazione logica è pervenire, a partire dallo schema concettuale, a uno schema logico che lo rappresenti **in modo fedele** e che sia, al tempo stesso, **“efficiente”**
 - L'**efficienza** è legata alle **prestazioni**, ma poiché **queste non sono valutabili precisamente a livello concettuale e logico** si deve far ricorso a degli **indicatori semplificati** per poter confrontare tra loro diverse alternative di traduzione
- Per quanto detto dividiamo il lavoro in due parti:
 - Progettazione logica **“fedele”**
 - Progettazione logica **“efficiente”**
- In ogni caso, a livello logico non introduciamo nuova informazione, con la possibile eccezione di qualche ridondanza (ma questa non è informazione “nuova”)

Progettazione logica “fedele” = equivalenza

Cosa si intende precisamente quando si dice che uno schema relazionale DB_{rel} rappresenta “fedelmente” uno schema concettuale (E/R) DB_{conc} ?

- Intuitivamente “fedeltà” vuol dire che mediante DB_{rel} possiamo rappresentare esattamente le stesse informazioni che possiamo rappresentare con DB_{conc} (*possiamo memorizzare gli stessi dati*)
- Più precisamente “fedeltà” significa che **i due schemi sono equivalenti dal punto di vista della loro capacità informativa**
- Il concetto di capacità informativa ha diverse definizioni, ma per i nostri scopi può essere considerato equivalente all’insieme delle istanze legali di uno schema, indicato con $Sat(DB)$
- Quindi

DB_{rel} e DB_{conc} sono equivalenti se $Sat(DB_{conc}) = Sat(DB_{rel})$

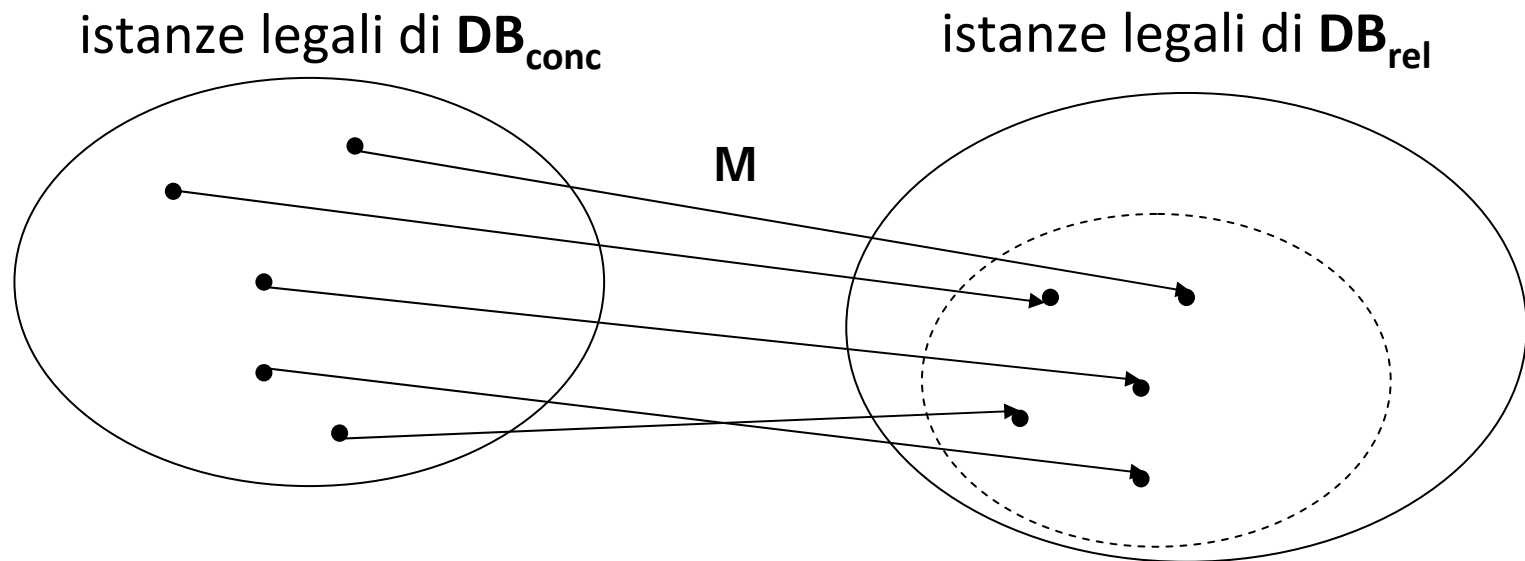
Progettazione che preserva l'informazione (1)

- Dato uno schema concettuale $\mathbf{DB}_{\text{conc}}$ consideriamo una progettazione che ne deriva uno schema logico-relazionale \mathbf{DB}_{rel}
- La nostra attività di progettazione può essere vista, a livello astratto, come la definizione di un **mapping** \mathbf{M} che ci dice come trasformare ogni istanza legale $\mathbf{db}_{\text{conc}}$ di $\mathbf{DB}_{\text{conc}}$ in una corrispondente istanza \mathbf{db}_{rel} di \mathbf{DB}_{rel}
- Diciamo che la progettazione **preserva l'informazione** se **\mathbf{M} è totale e iniettiva**:
 - (totale) per ogni istanza $\mathbf{db}_{\text{conc}}$ di $\mathbf{DB}_{\text{conc}}$ esiste un'istanza \mathbf{db}_{rel} di \mathbf{DB}_{rel} tale che $\mathbf{M}(\mathbf{db}_{\text{conc}}) = \mathbf{db}_{\text{rel}}$ e
 - (iniettiva) non esistono due istanze $\mathbf{db1}_{\text{conc}}$ e $\mathbf{db2}_{\text{conc}}$ tali che $\mathbf{M}(\mathbf{db1}_{\text{conc}}) = \mathbf{M}(\mathbf{db2}_{\text{conc}})$

Progettazione che preserva l'informazione (2)

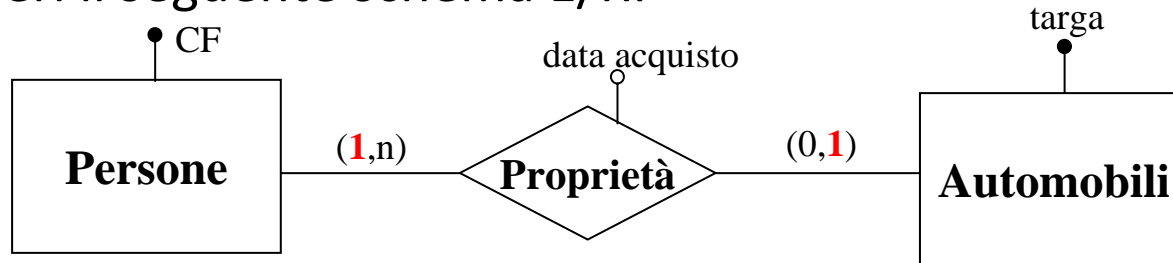
■ Preservare l'informazione:

- la definizione intuitivamente asserisce che lo schema relazionale può contenere i dati dello schema E/R (totalità) e che si può "tornare indietro" (iniettività)



Perché non basta

- Si consideri il seguente schema E/R:



e lo schema relazionale:

Persone(CF)
Auto(Targa)
Proprieta(CF, Targa, DataAcquisto)
FK: CF REFERENCES Persone
FK: Targa REFERENCES Auto

- La traduzione preserva l'informazione, ma **esistono infinite istanze che sono legali rispetto a DB_{rel} e che non lo sono per DB_{conc} !**

Persone

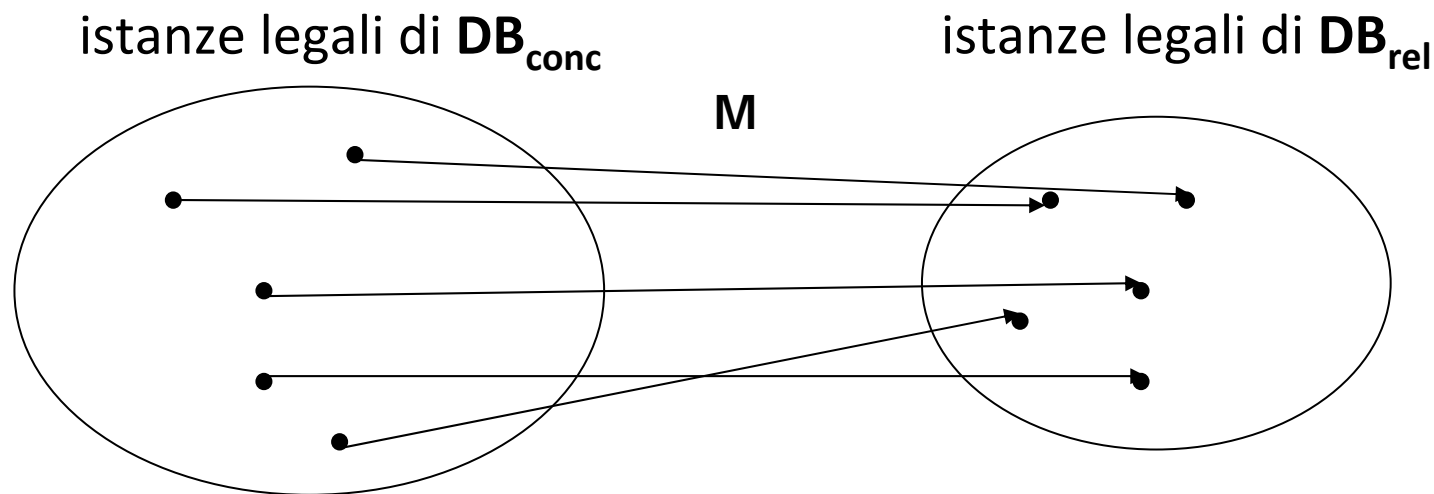
<u>CF</u>
BLGSTR71B22
FDLNNR66M45
BSZNTN82L27

<u>CF</u>	<u>Targa</u>	DataAcquisto
BLGSTR71B22	CT 001 MJ	12/08/2004
FDLNNR66M45	CT 001 MJ	15/07/2003

Proprieta

Progettazione che garantisce l'equivalenza

- Diciamo che la progettazione **garantisce l'equivalenza** se:
 - preserva l'informazione, e
 - per ogni istanza legale \mathbf{db}_{rel} di \mathbf{DB}_{rel} esiste un'istanza legale \mathbf{db}_{conc} di \mathbf{DB}_{conc} tale che $\mathbf{M}(\mathbf{db}_{conc}) = \mathbf{db}_{rel}$
- La definizione intuitivamente asserisce che esiste una **biiezione** tra gli insiemi di istanze legali

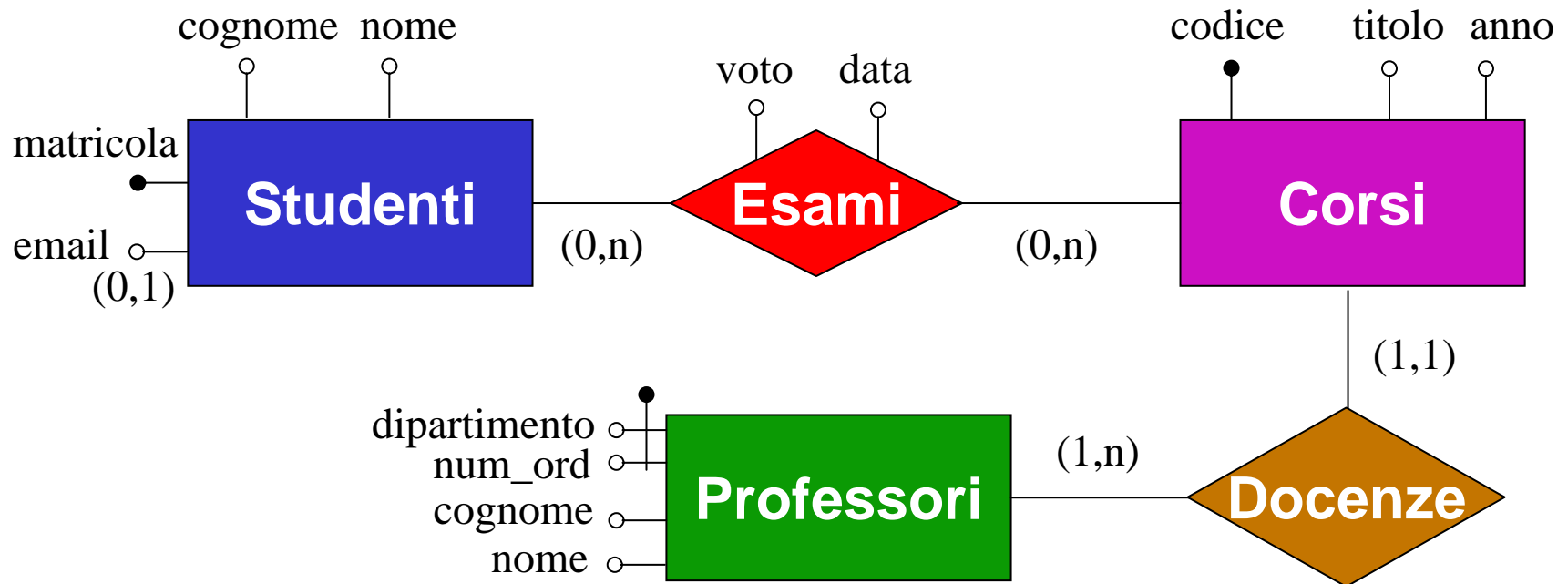


In pratica

- La definizione data di equivalenza non è “operativa”, in quanto non dice nulla su come debba essere fatta una traduzione che garantisca l’equivalenza degli schemi
- Tuttavia può essere usata “localmente”:
In pratica, la traduzione avviene operando una **sequenza di trasformazioni/traduzioni semplici**, per ognuna delle quali è altrettanto semplice dare delle regole che garantiscono l’equivalenza
- Per quanto visto, possiamo dividere queste regole in:
 - **regole che preservano l’informazione (regole sulla “struttura”)**
 - **regole aggiuntive che garantiscono l’equivalenza (regole sui vincoli)**
- Vedremo che l’equivalenza può solo essere in parte garantita dal DDL di SQL, e quindi c’è bisogno di altro...

Traduzione di schemi E/R “semplici”

- Consideriamo il caso più semplice in cui:
 - Abbiamo entità e associazioni, ma **non gerarchie**
 - Ogni entità ha **un singolo identificatore**, ed è **interno**
 - Non abbiamo **attributi ripetuti**



Traduzione di base: entità

- Ogni entità è tradotta con una relazione con gli stessi attributi
 - La **chiave primaria** coincide con l'**identificatore** dell'entità
 - Se un **attributo** è **opzionale** permettiamo la presenza di **valori nulli**, e usiamo l'asterisco (*) per indicare tale possibilità

Studenti

<u>Matricola</u>	Cognome	Nome	Email*
29323	Bianchi	Giorgio	gbianchi@alma.unibo.it
35467	Rossi	Anna	
39654	Verdi	Marco	
42132	Neri	Lucia	lucia78@cs.ucsd.edu

Corsi

<u>Codice</u>	Titolo	Anno
483	Analisi	1
729	Analisi	1
815	Elettronica	2
913	Sistemi Informativi	3

Professori

<u>Dip</u>	<u>Num_ord</u>	Cognome	Nome
DEIS	12	Bolzano	Bernhard
DEIS	15	Codd	Ted
DIE	12	Marconi	Guglielmo

Traduzione di base: associazioni

- 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 primaria** dipende dalle **cardinalità massime** delle entità nell'associazione

Esami

<u>Matricola</u>	<u>Codice</u>	Voto	Data
29323	483	28	12/06/2003
39654	729	30	15/07/2003
29323	913	26	12/06/2003
35467	913	30	20/09/2004

Docenze

<u>Codice</u>	Dip	Num_ord
483	DEIS	12
729	DEIS	12
815	DIE	12
913	DEIS	15

Nomi delle foreign key

- Non è ovviamente necessario mantenere per le foreign key gli stessi nomi delle primary key referenziate, conviene usare nomi più espressivi

```
Esami(Matricola, CodCorso, Voto, Data)
```

```
FK: Matricola REFERENCES Studenti
```

```
FK: CodCorso REFERENCES Corsi
```

```
Docenze(CodCorso, Dip, Num_ordine)
```

```
FK: CodCorso REFERENCES Corsi
```

```
FK: (Dip, Num_ordine) REFERENCES Professori
```

- E' evidente che se le entità collegate hanno un identificatore con lo stesso nome la ridenominazione è obbligatoria!

Associazioni molti a molti

- La **chiave primaria** coincide con l'unione degli **identificatori delle entità** collegate

Esami (**Matricola**, **CodCorso**, **Voto**, **Data**)

FK: Matricola REFERENCES Studenti

FK: CodCorso REFERENCES Corsi

- Per le associazioni molti a molti la traduzione presentata, ovvero con una relazione a sé, è la sola alternativa ragionevole
 - Vedremo alternative “non ragionevoli” quando parleremo di normalizzazione di schemi relazionali

Associazioni uno a molti

- La **chiave primaria** coincide con l'**identificatore dell'entità che partecipa con cardinalità massima 1**

Docenze(**CodCorso**,Dip,Num_ordine)

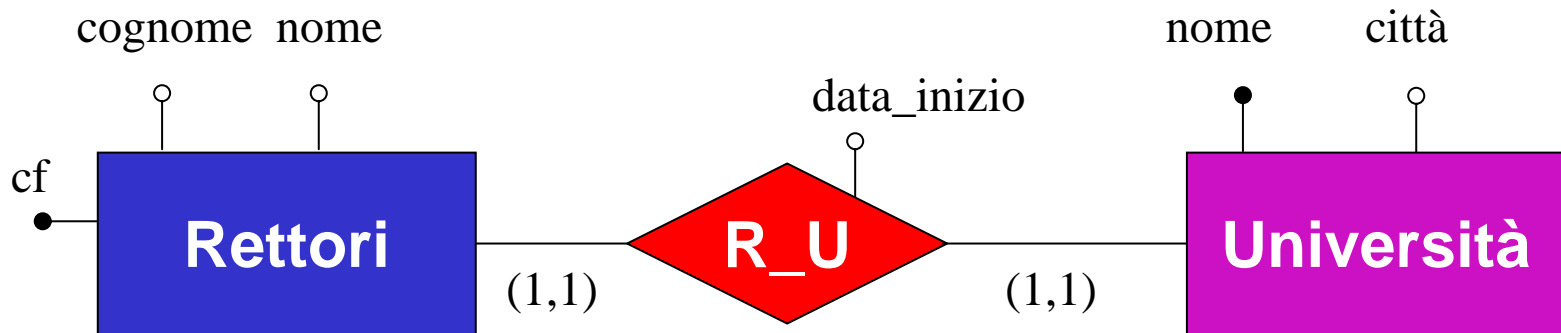
FK: CodCorso REFERENCES Corsi

FK: (Dip,Num_ordine) REFERENCES Professori

- Perché?
- Poiché ogni corso c ha al massimo 1 docente, allora c può comparire al massimo una volta in Docenze

Associazioni uno a uno

- La **chiave primaria** è **uno** dei due identificatori, l'**altro** diventa una **chiave alternativa**



R_U(CF, NomeUniversità)

FK: CF REFERENCES Rettori

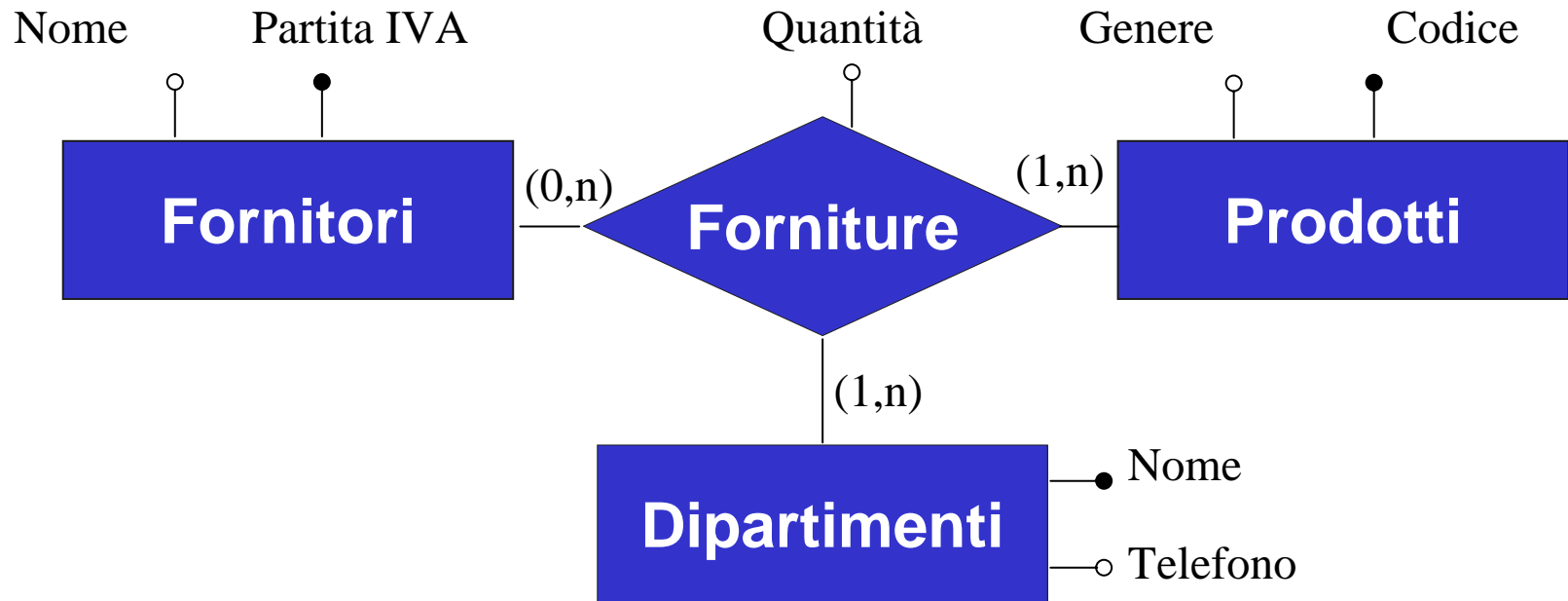
FK: NomeUniversità REFERENCES Università

Unique(NomeUniversità)

La scelta dipende
dall'**importanza relativa**
delle chiavi

Associazioni n-arie

- Valgono le considerazioni già fatte, ad esempio:



Fornitori(PartitaIVA, Nome)

Prodotti(Codice, Genere)

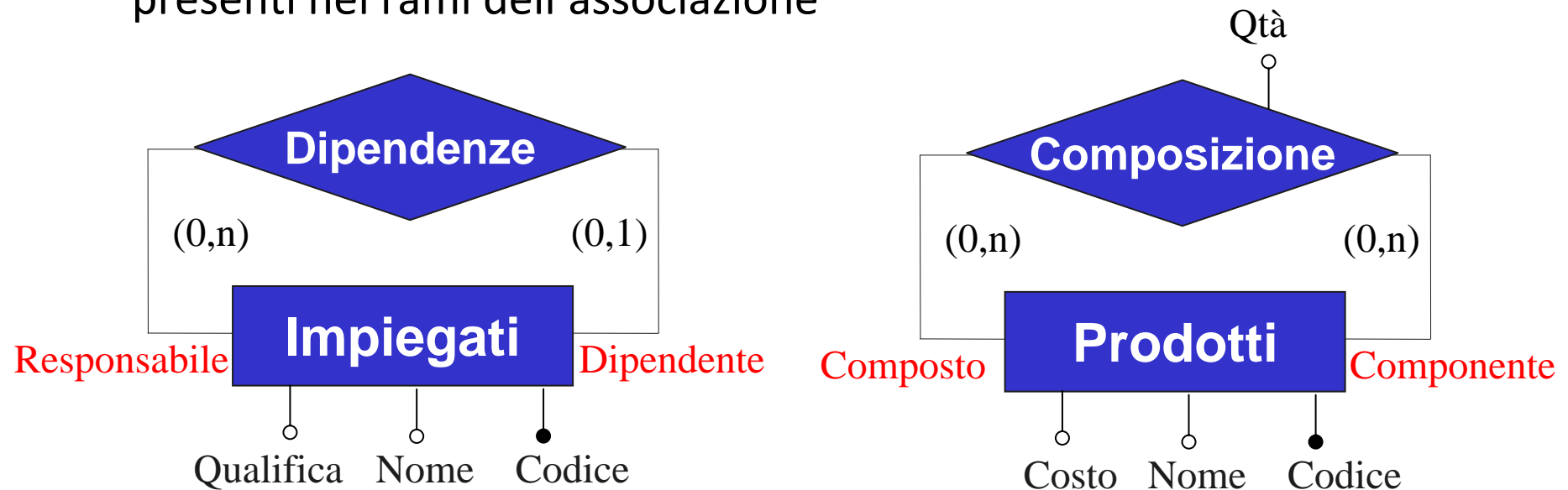
Dipartimenti(Nome, Telefono)

Forniture(Fornitore, Prodotto, Dipartimento, Quantità)

FK: ...

Associazioni ad anello

- In questo caso i **nomi delle foreign key** si possono derivare dai **ruoli** presenti nei rami dell'associazione



```
Impiegati(Codice, Nome, Qualifica)
Dipendenze(Dipendente, Responsabile)
FK: Dipendente REFERENCES Impiegati
FK: Responsabile REFERENCES Impiegati
```

```
Prodotti(Codice, Nome, Costo)
Composizione(Composto, Componente, Qta)
FK: Composto REFERENCES Prodotti
FK: Componente REFERENCES Prodotti
```

Associazioni uno a molti: alternative (1)

- Per le associazioni uno a molti è possibile considerare anche una traduzione più compatta, che ingloba l'associazione nella relazione che traduce l'entità partecipante con cardinalità massima 1

Corsi

<u>Codice</u>	Titolo	Anno
483	Analisi	1
729	Analisi	1
815	Elettronica	2
913	Sistemi Informativi	3

Docenze

<u>Codice</u>	Dip	Num_ordine
483	DEIS	12
729	DEIS	12
815	DIE	12
913	DEIS	15



Corsi

<u>Codice</u>	Titolo	Anno	Dip_docente	Num_ordine_docente
483	Analisi	1	DEIS	12
729	Analisi	1	DEIS	12
815	Elettronica	2	DIE	12
913	Sistemi Informativi	3	DEIS	15

Associazioni uno a molti: alternative (2)

- Il vantaggio che se ne trae si apprezza in fase di interrogazione:
 - Si semplifica la scrittura di query SQL e, soprattutto,
 - Si riduce il numero di join da eseguire (migliori prestazioni!)

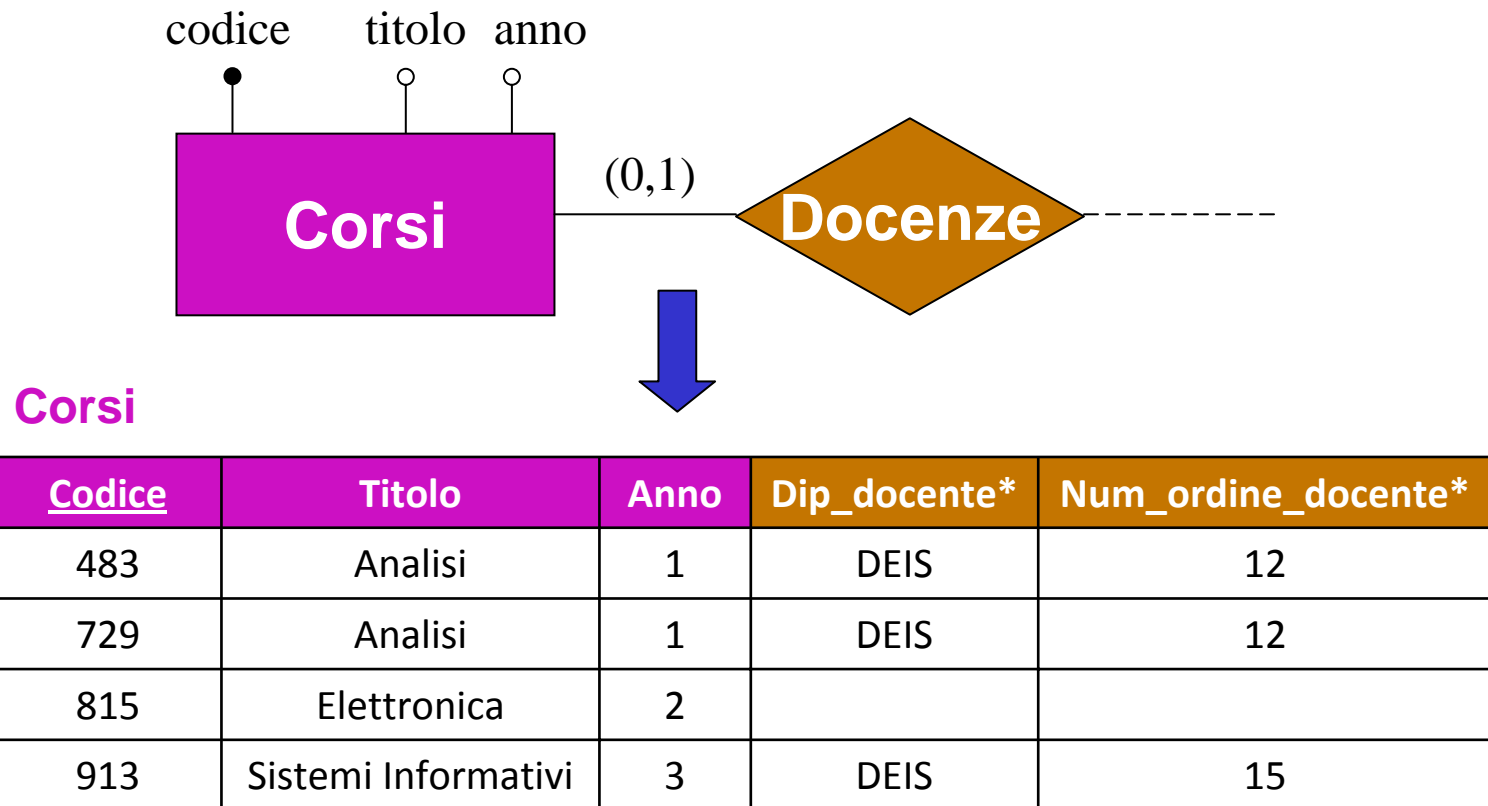
“Nome e cognome degli insegnanti di Analisi”

```
SELECT  P.Nome, P.Cognome
FROM    Corsi C, Docenze D, Professori P
WHERE   C.Codice = D.CodCorso
AND     D.Dip = P.Dip AND D.Num_ordine = P.Num_ord
AND     C.Titolo = 'Analisi'
```

```
SELECT  P.Nome, P.Cognome
FROM    Corsi C, Professori P
WHERE   C.Dip_docente = P.Dip
AND     C.Num_ordine_docente = P.Num_ord
AND     C.Titolo = 'Analisi'
```

Associazioni uno a molti: alternative (3)

- Per contro, bisogna stare attenti se **l'entità che ingloba l'associazione partecipa con cardinalità minima 0**
- Supponiamo che un corso possa anche non avere un docente...



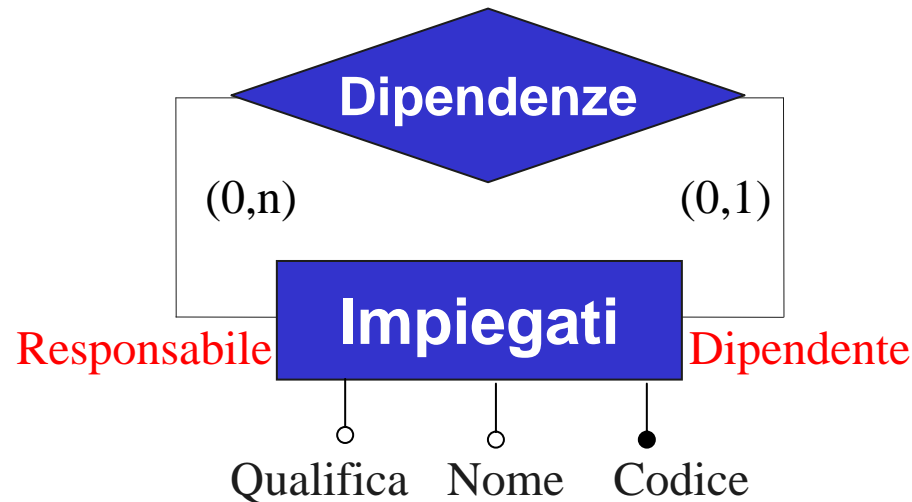
Associazioni uno a molti: alternative (4)

- La presenza dei valori nulli può essere tollerata se questi sono relativamente pochi (*dipende dai volumi dei dati in gioco*), altrimenti si ha uno spreco inutile di spazio!
- **Bisogna ovviamente garantire che i valori della foreign key siano o tutti definiti o tutti nulli**

```
Corsi(Codice, Titolo, Anno, Dip_docente*, Num_ordine_docente*)  
  FK: (Dip_docente, Num_ordine_docente) REFERENCES Professori  
  CHECK (  
    (Dip_docente IS NOT NULL AND Num_ordine_docente IS NOT NULL)  
    OR  
    (Dip_docente IS NULL AND Num_ordine_docente IS NULL)  
  )
```

Associazioni uno a molti ad anello

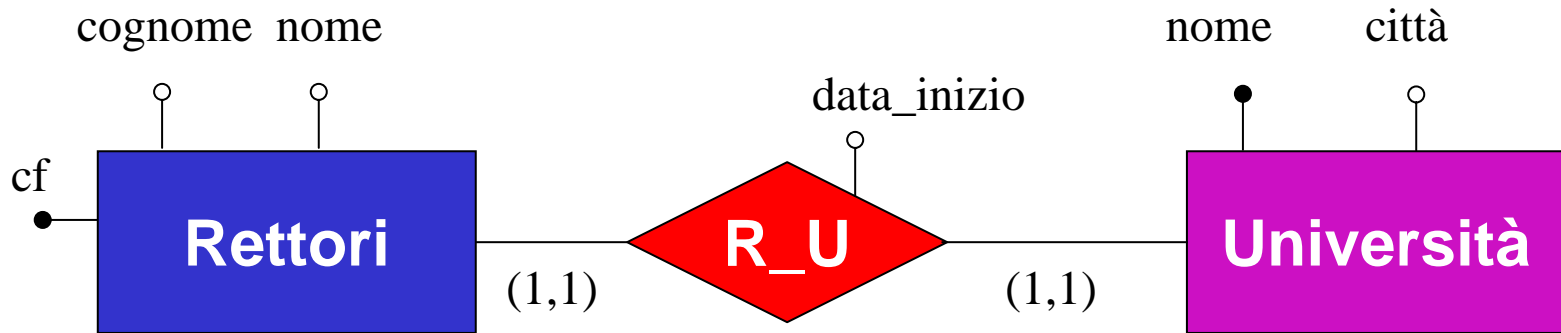
- Quanto visto si applica anche alle associazioni ad anello, ad esempio:



```
Impiegati(Codice, Nome, Qualifica, CodiceResponsabile*)  
FK: CodiceResponsabile REFERENCES Impiegati
```

Associazioni uno a uno: alternative (1)

- Si hanno a disposizione varie possibilità:



`Rettori(CF, Cognome, Nome, NomeUniversità, DataInizio)`

`FK: NomeUniversità REFERENCES Università`

`Unique(NomeUniversità)`

oppure

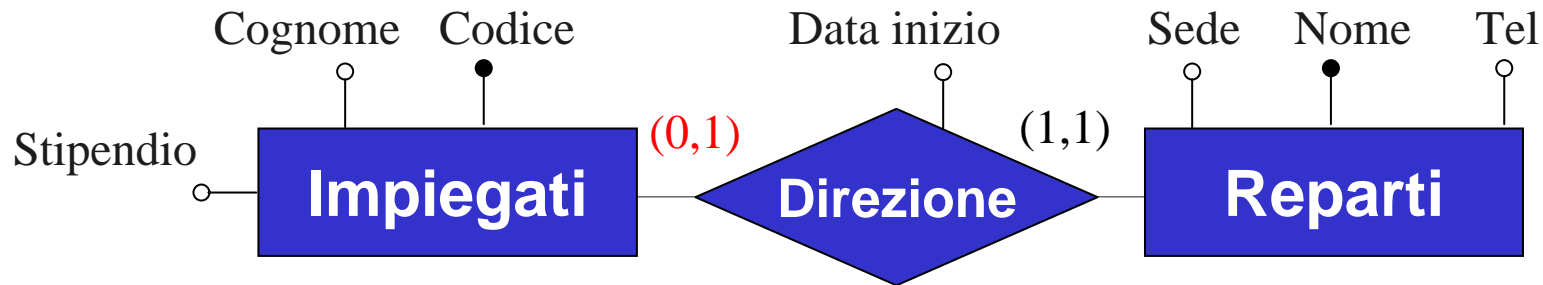
`Università(Nome, Città, CFRettore, DataInizio)`

`FK: CFRettore REFERENCES Rettori`

`Unique(CFRettori)`

Associazioni uno a uno: alternative (2)

- Se $\text{min-card}(E,R) = 0$, tradurre l'associazione R inglobandola in E non è in generale una buona scelta (dipende dai volumi dei dati in gioco)



Impiegati(Codice, Cognome, Stipendio, **Reparto***, **DataInizio***)

FK: Reparto REFERENCES Reparti

Unique(Reparto)

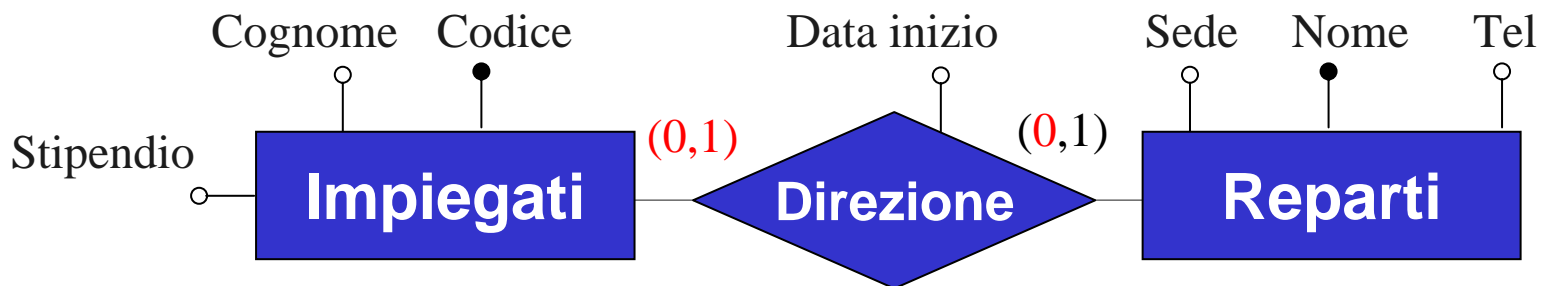
CHECK ((Reparto IS NOT NULL AND DataInizio IS NOT NULL) OR
(Reparto IS NULL AND DataInizio IS NULL))

Reparto(Nome, Sede, Telefono)

MOLTI VALORI NULLI!!

Associazioni uno a uno: alternative (3)

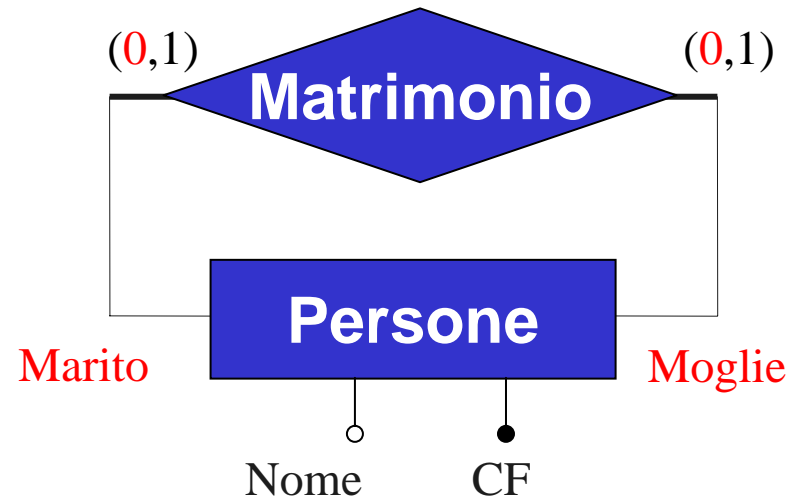
- E' possibile anche operare una traduzione in cui le entità vengono "accorpate". In ogni caso:
 - Se $\text{min-card}(E1,R) = \text{min-card}(E2,R) = 1$ si avranno **due chiavi, entrambe senza valori nulli** (la chiave primaria è "la più importante")
 - Se $\text{min-card}(E1,R) = 0$ e $\text{min-card}(E2,R) = 1$ la chiave derivante da E2 ammetterà valori nulli, e **la chiave primaria si ottiene da E1**
 - Se $\text{min-card}(E1,R) = \text{min-card}(E2,R) = 0$ entrambe le chiavi hanno valori nulli, quindi si rende necessario **introdurre un codice**



ImpRep(CodiceImpDip, CodiceImp*, ..., Reparto*, ..., DataInizio*)

Associazioni uno a uno ad anello

- In questo caso è possibile ancora inglobare l'associazione nell'entità
- La traduzione è ancora problematica se entrambe le partecipazioni sono opzionali



1 relazione:

`Persone (Codice , CFUomo* , NomeUomo* , CFDonna* , NomeDonna*)`

2 relazioni:

`Persone (CF , Nome)`

`Matrimoni (CFMarito , CFMoglie) -- o Matrimoni (CFMoglie , CFMarito)`

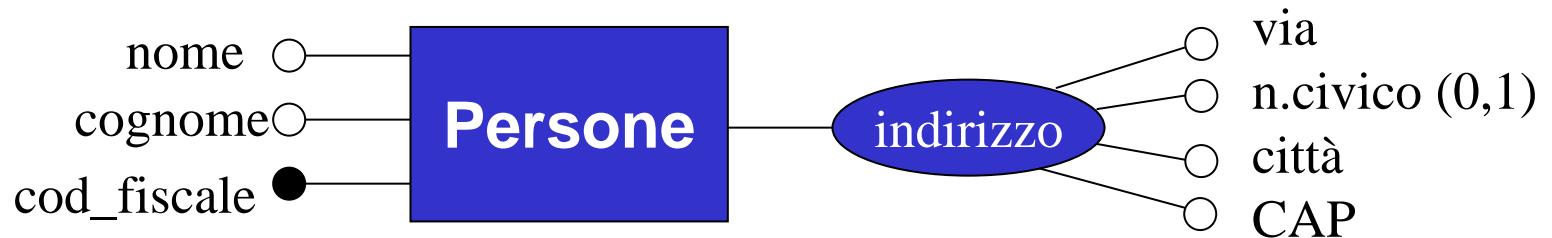
`FK: CFMarito REFERENCES Persone`

`FK: CFMoglie REFERENCES Persone`

`Unique (CFMoglie)`

Attributi composti

- Gli **attributi composti** vengono tradotti suddividendoli ricorsivamente nelle loro componenti
- In alternativa si possono mappare in un singolo attributo della relazione, il cui dominio va opportunamente definito, ma questa scelta ovviamente porta a un impoverimento della rappresentazione
- E' consigliabile usare un **prefisso comune** per gli attributi che si ottengono



Persone (**CF**, **Cognome**, **Nome**, **Ind_Via**, **Ind_NCivico***, **Ind_Città**, **Ind_CAP**)

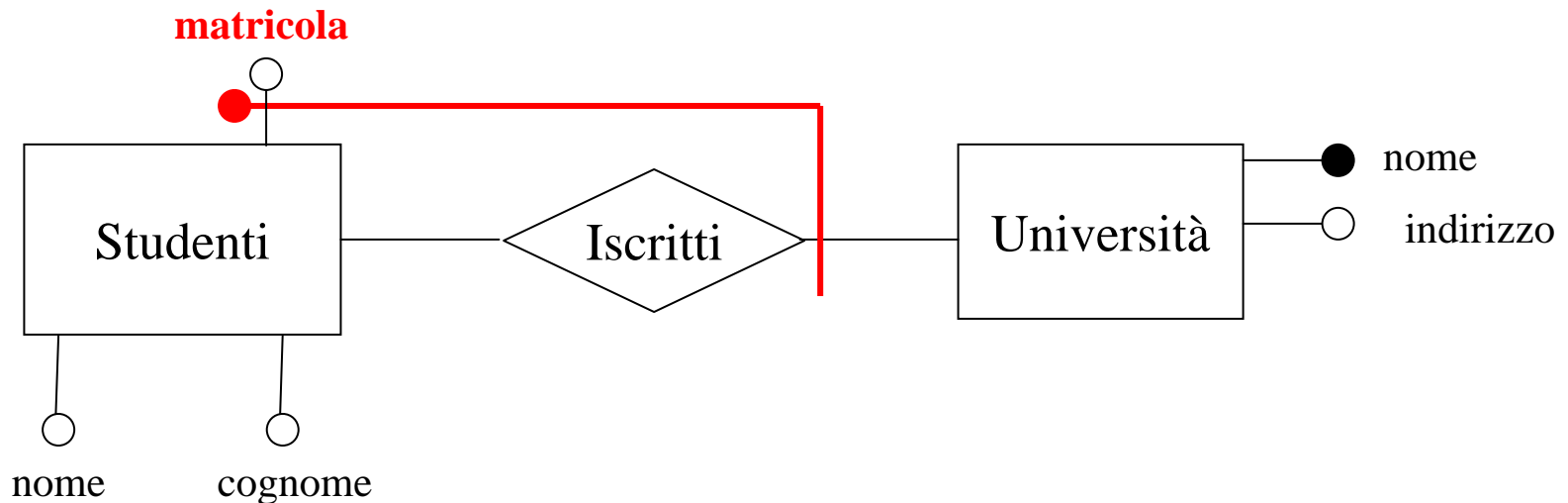
Equivalenza?

- Tutte le trasformazioni viste preservano l'informazione
- Per quanto riguarda l'equivalenza, viceversa, anche con schemi E/R semplici nascono dei problemi
- In particolare, quando $\text{min-card}(E,A) = 1$ e l'associazione E non viene tradotta inglobandola in E (cosa non possibile se $\text{max-card}(E,A) = n$), non c'è modo con il DDL di SQL supportato dai DBMS di garantire la totalità della partecipazione
- Infatti, se la chiave primaria di E è K, occorrerebbe aggiungere nella definizione relazionale di E il vincolo (generato da DB-MAIN!):

```
CONSTRAINT min_card_E_A_1  
CHECK (EXISTS (SELECT * FROM A WHERE A.K = K))
```

Entità con identificazione esterna

- Nel caso di entità identificata esternamente, si “importa” l’identificatore della/e entità identificante/i.
- L’associazione relativa risulta automaticamente tradotta



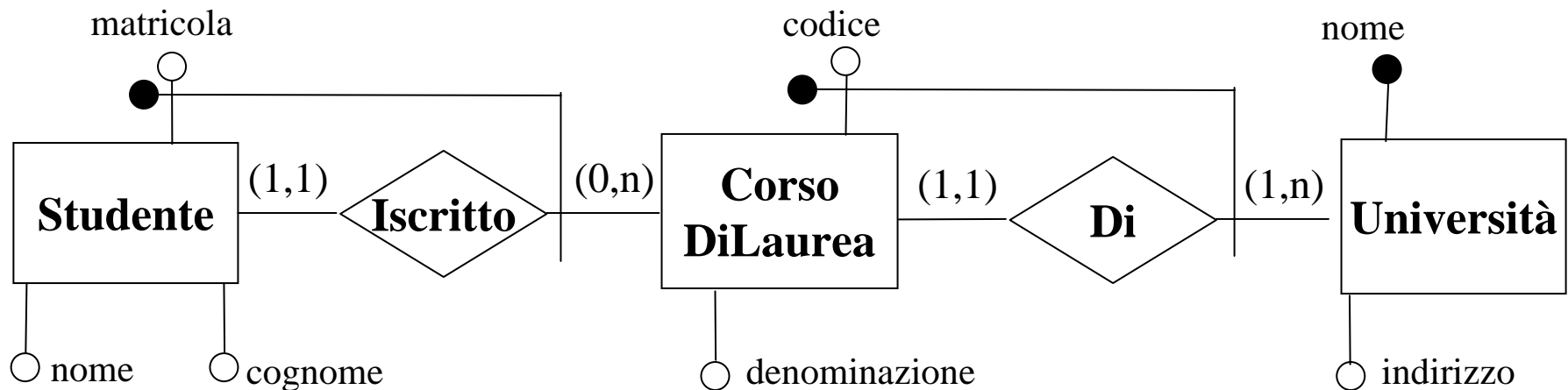
`Studente(Matricola, NomeUniversità, Cognome, Nome)`

`FK: NomeUniversità REFERENCES Università`

`Università(Nome, Indirizzo)`

Identificazioni esterne: una precisazione

- Nel caso generale, si possono avere **identificazioni esterne in cascata**
- Per operare correttamente occorre **partire dalle entità non identificate esternamente** e propagare gli identificatori che così si ottengono



Università(Nome, Indirizzo)

CorsoDiLaurea(NomeUniversità, Codice, Denominazione)

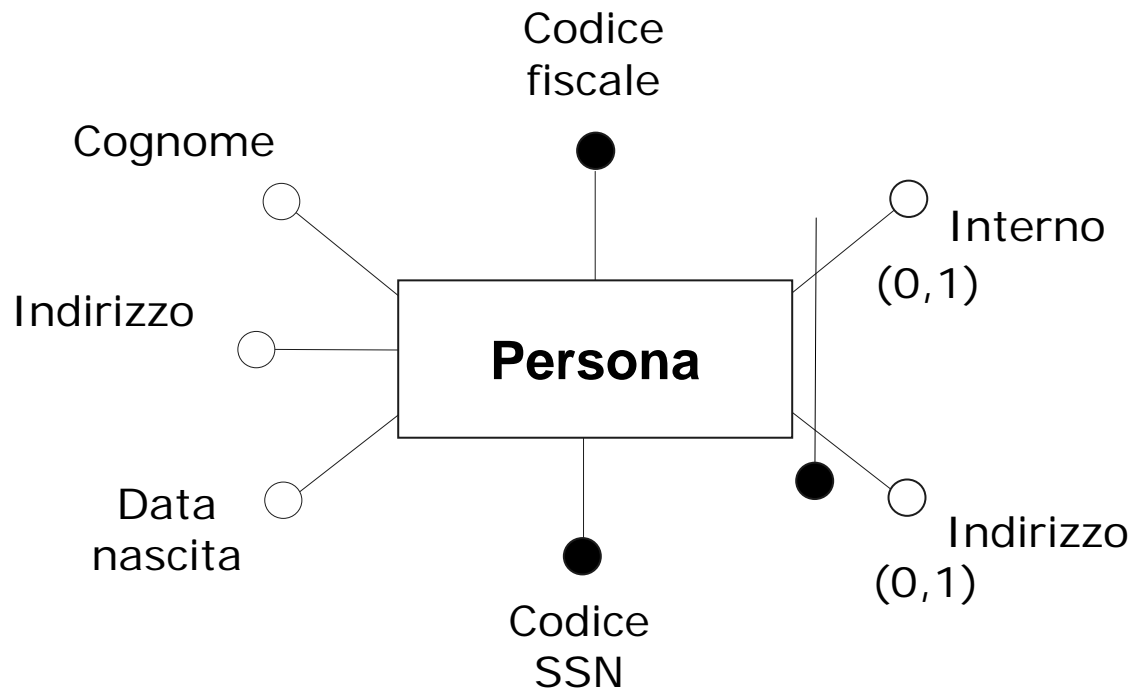
Studente(NomeUniversità, CodiceCdL, Matricola, Cognome, Nome)

Entità con più identificatori

- Nel caso di più identificatori si pone il problema di quale scegliere per generare la corrispondente **chiave primaria**
- I criteri da adottare sono:
 - **assenza di opzionalità** (valori NULL)
 - **semplicità**
 - **utilizzo nelle operazioni più frequenti o importanti**
- Se nessuno degli identificatori soddisfa i requisiti si introducono dei nuovi attributi (dei “**codici**”) allo scopo

Identificatori principali: esempio

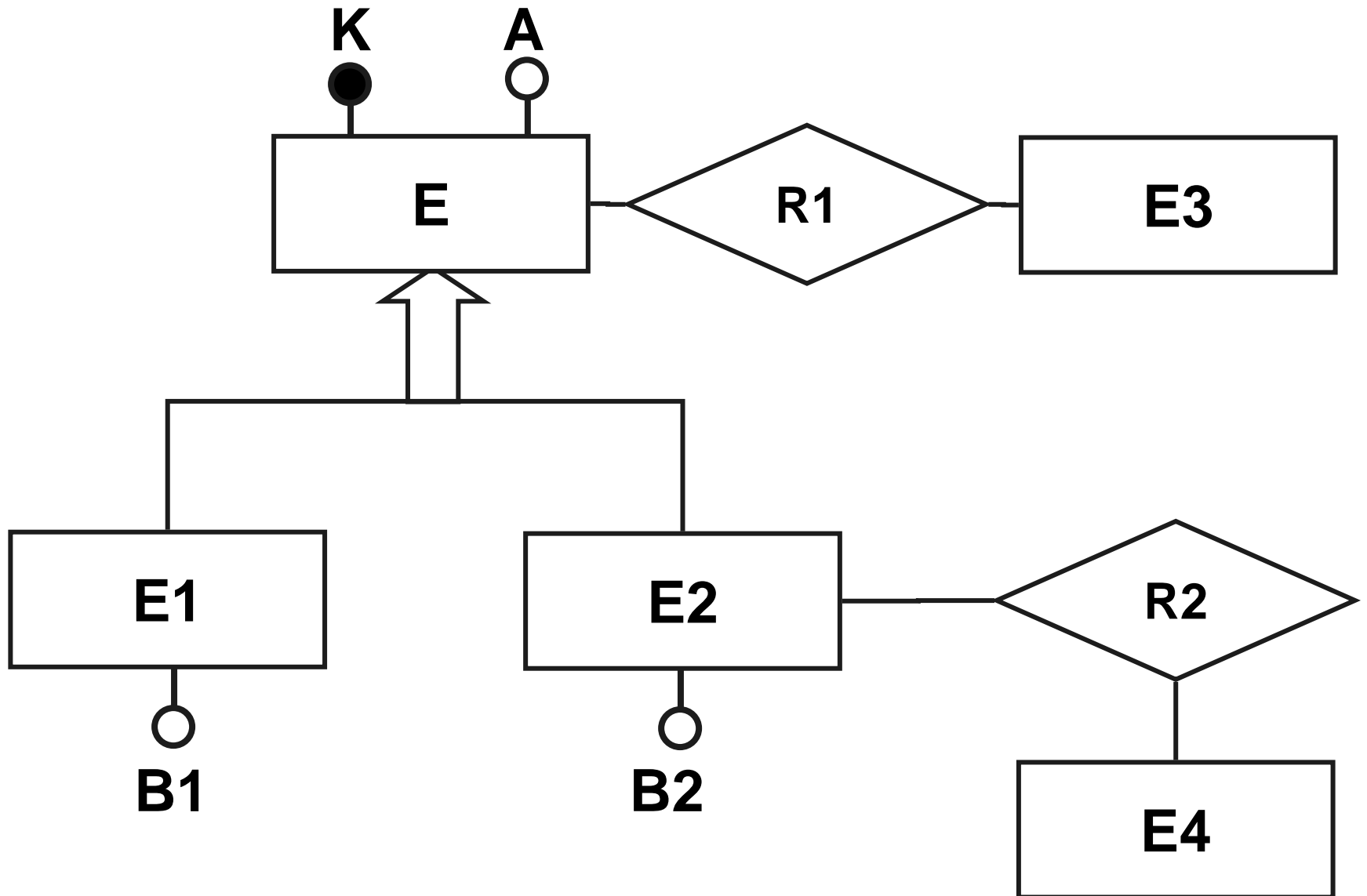
- L'identificatore {Interno, Indirizzo} è opzionale, quindi non può essere scelto
- Tra Codice fiscale e Codice SSN la scelta dipende da quale è più "importante" (frequentemente usato) per accedere a una persona



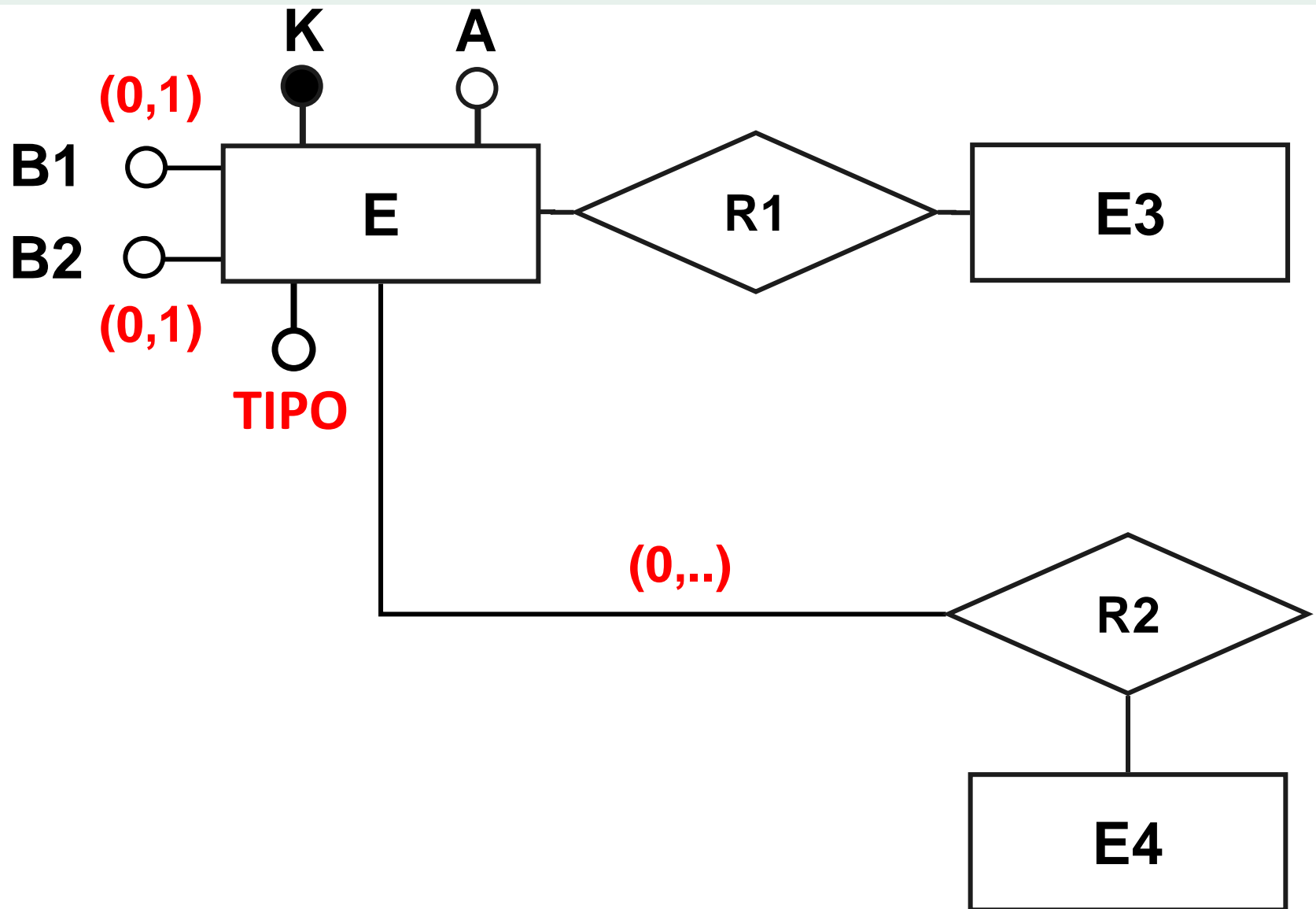
Eliminazione delle gerarchie

- Il modello relazionale non può rappresentare direttamente le generalizzazioni
- Si eliminano perciò le gerarchie, sostituendole con entità e associazioni
- Vi sono 3 possibilità (più altre soluzioni intermedie, o “ibride”):
 - Accorpare le entità figlie nel genitore (collasso verso l’alto)
 - Accorpare il genitore nelle entità figlie (collasso verso il basso)
 - Sostituire la generalizzazione con associazioni (traduzione indipendente)
- Poiché sappiamo già tradurre entità e associazioni, possiamo anche ragionare sugli schemi relazionali che ne derivano

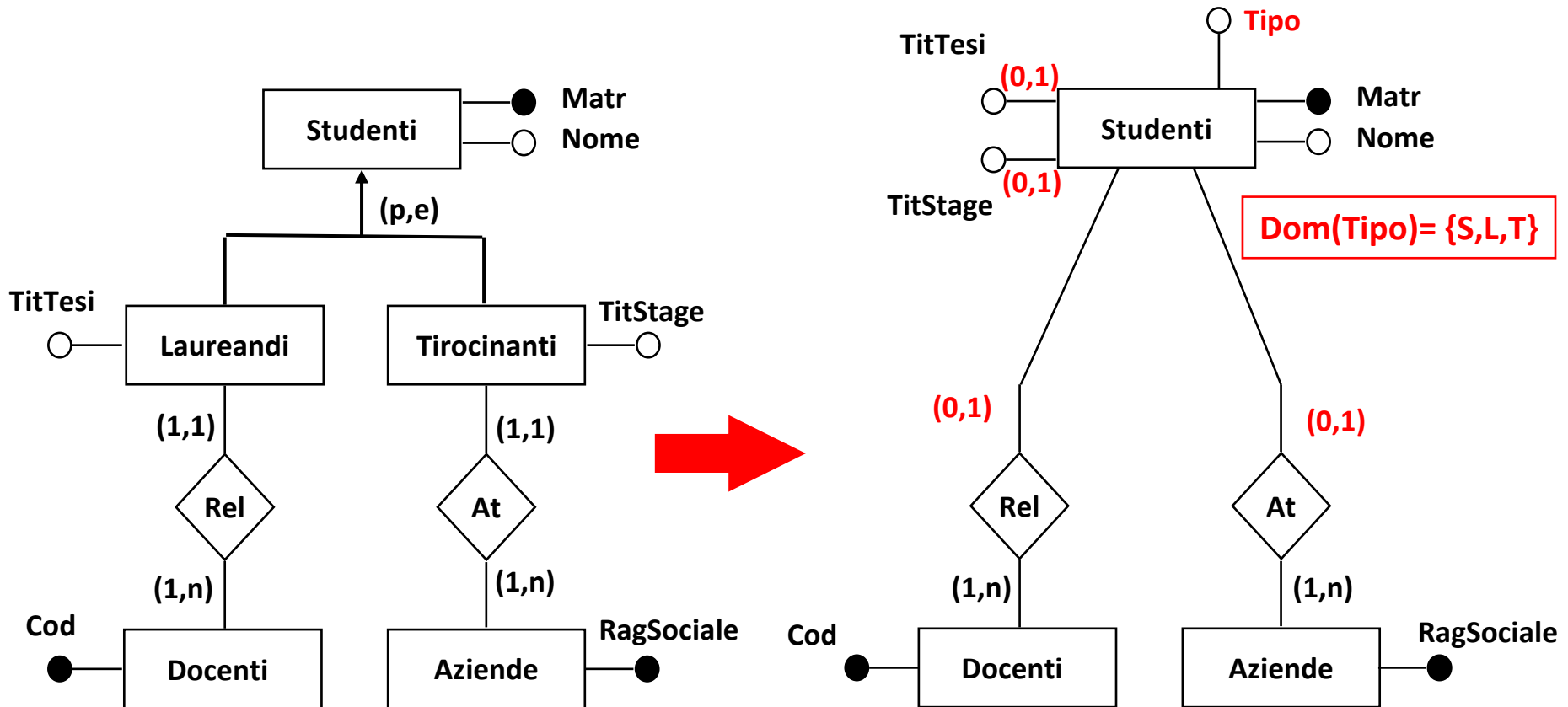
Schema di riferimento



1. Accorpare nel genitore...



1. Esempio



1. Esempio: traduzione relazionale

- A livello logico si genera una singola relazione, ed è necessario inserire un vincolo che “sincronizzi” i valori di Tipo con quelli degli attributi delle entità figlie (ora opzionali):

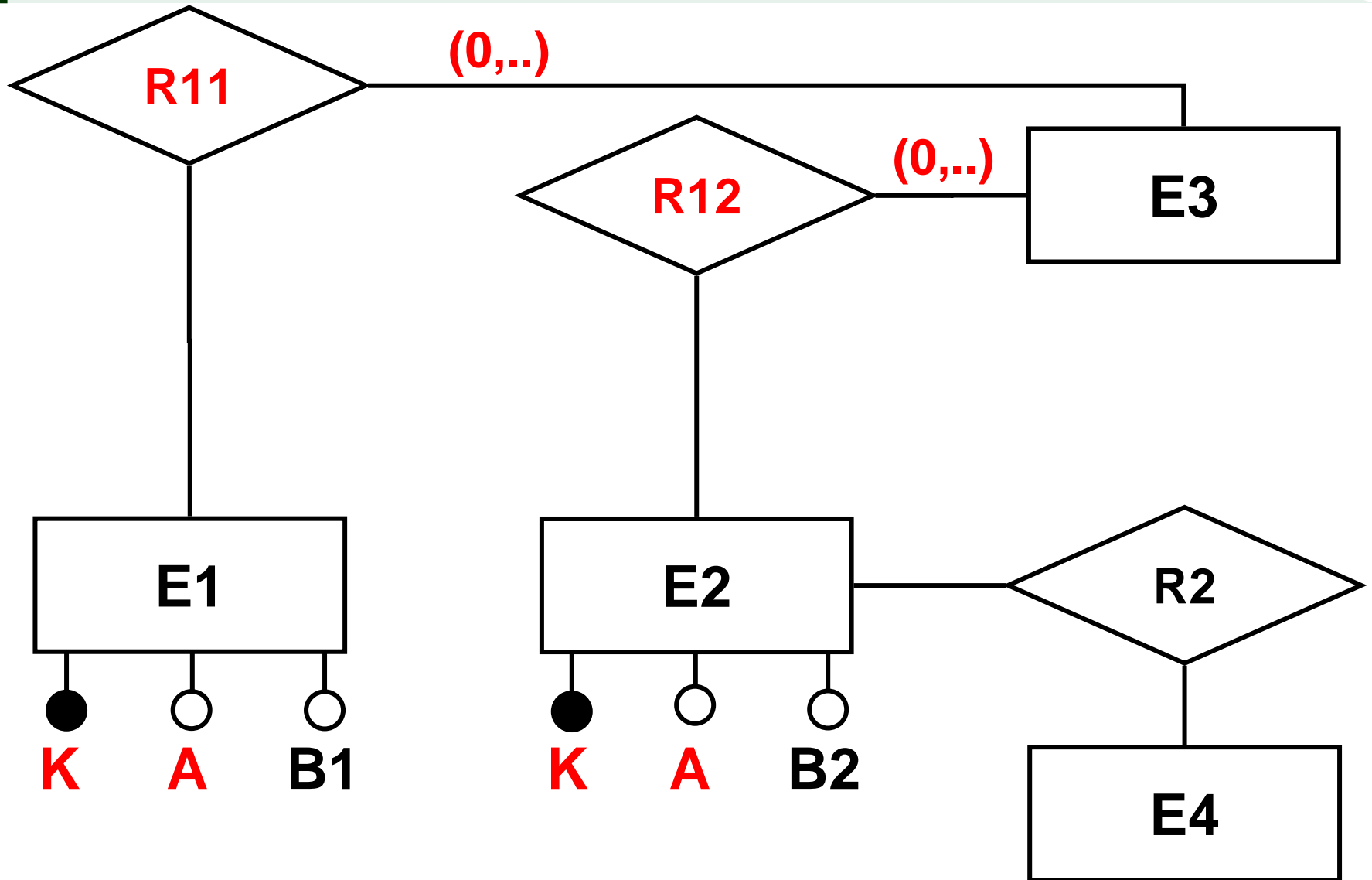
```
Studenti ( Matr, Nome, Tipo, TitTesi*, TitStage* )  
CHECK (  
    (Tipo = 'S' AND TitTesi IS NULL AND TitStage IS NULL)  
    OR (Tipo = 'L' AND TitTesi IS NOT NULL AND TitStage IS NULL)  
    OR (Tipo = 'T' AND TitTesi IS NULL AND TitStage IS NOT NULL)  
)
```

- Se le associazioni cui partecipavano le entità figlie si inglobano in Studenti, il CHECK deve coinvolgere anche le relative foreign key
- Altrimenti è comunque necessario garantire che, ad es., ogni Laureando abbia un Relatore

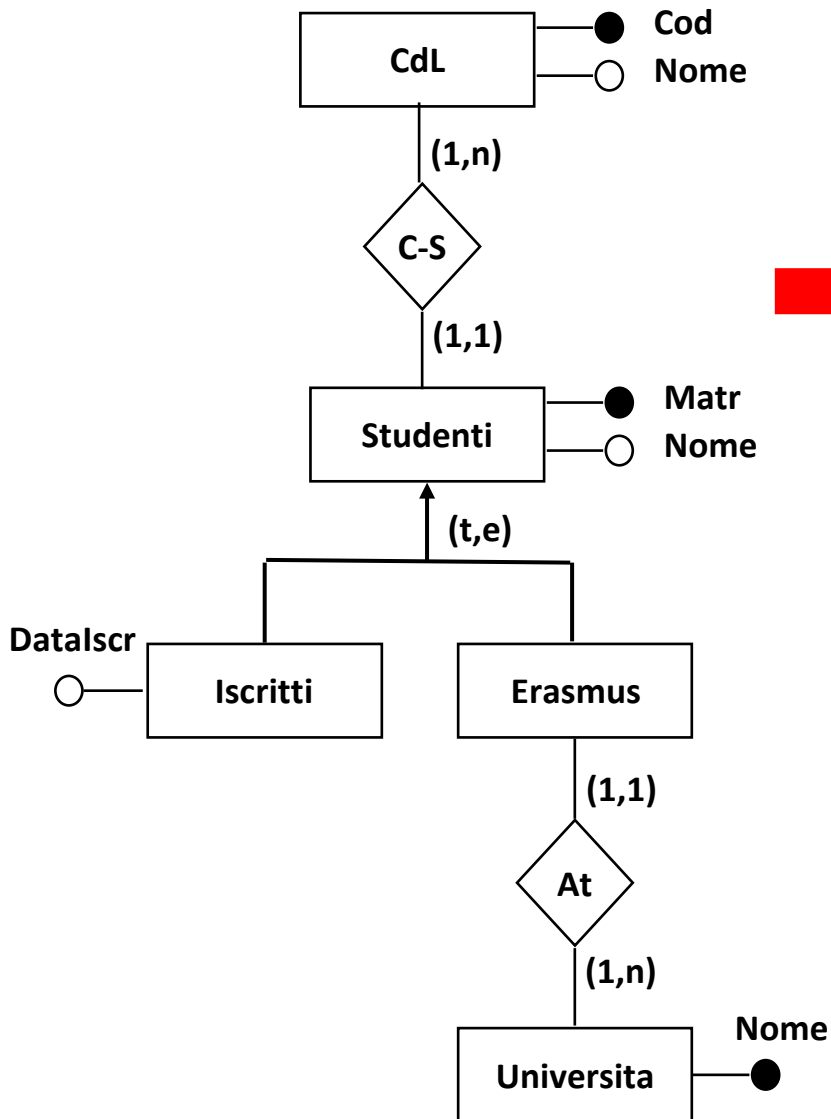
Collasso verso l'alto: osservazioni

- Nell'esempio, **Tipo** è un attributo **selettore** che specifica se una istanza di E appartiene (anche) a una delle N entità figlie
- La soluzione da adottare dipende dalla **copertura della gerarchia**:
 - **totale esclusiva**: Tipo ha N valori, quante sono le entità figlie
 - **parziale esclusiva**: Tipo ha N+1 valori; il valore in più serve per le istanze che non appartengono a nessuna entità figlia
 - **sovrapposta**: occorrono tanti selettori quante sono le entità figlie, ciascuno a valore booleano: Tipo_i è vero per ogni istanza di E che appartiene (anche) 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 entità figlie si trasportano su E, e le cardinalità minime diventano tutte 0

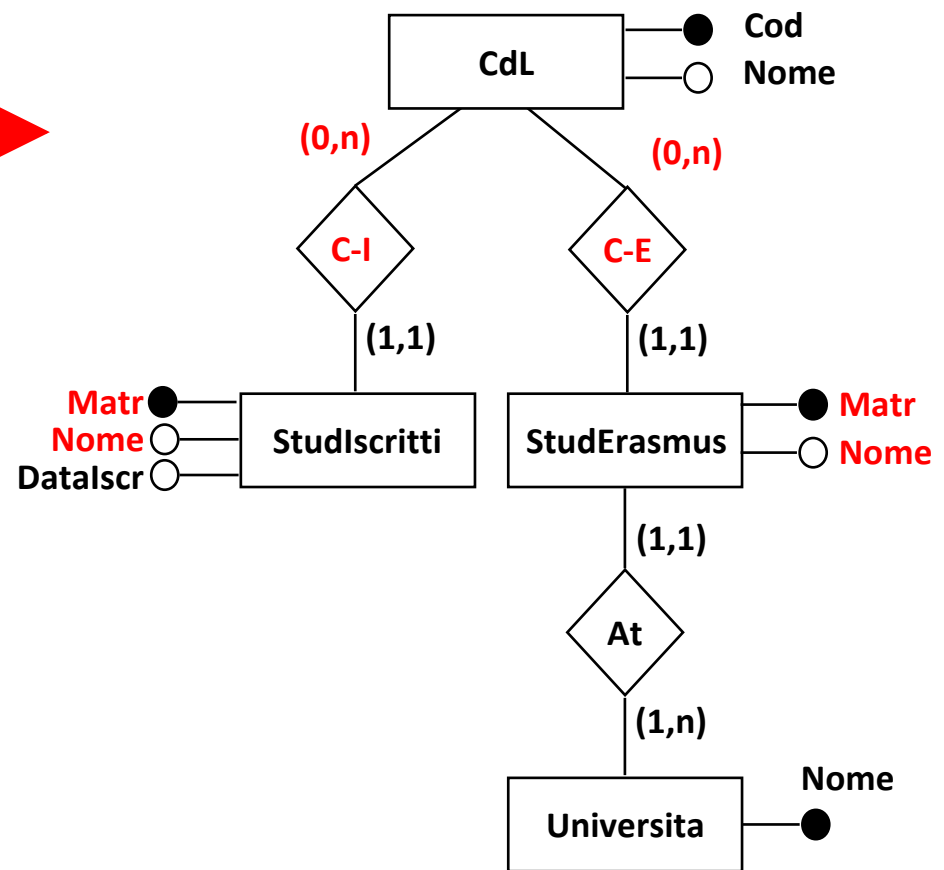
2. Accorpare nelle figlie...



2. Esempio



Prog. Logica: traduzione



Sistemi Informativi T

2. Esempio: traduzione relazionale

- A livello logico si generano N relazioni, una per ogni entità figlia

```
StudIscritti(Matr, Nome, DataIscriz)  
StudErasmus(Matr, Nome)
```

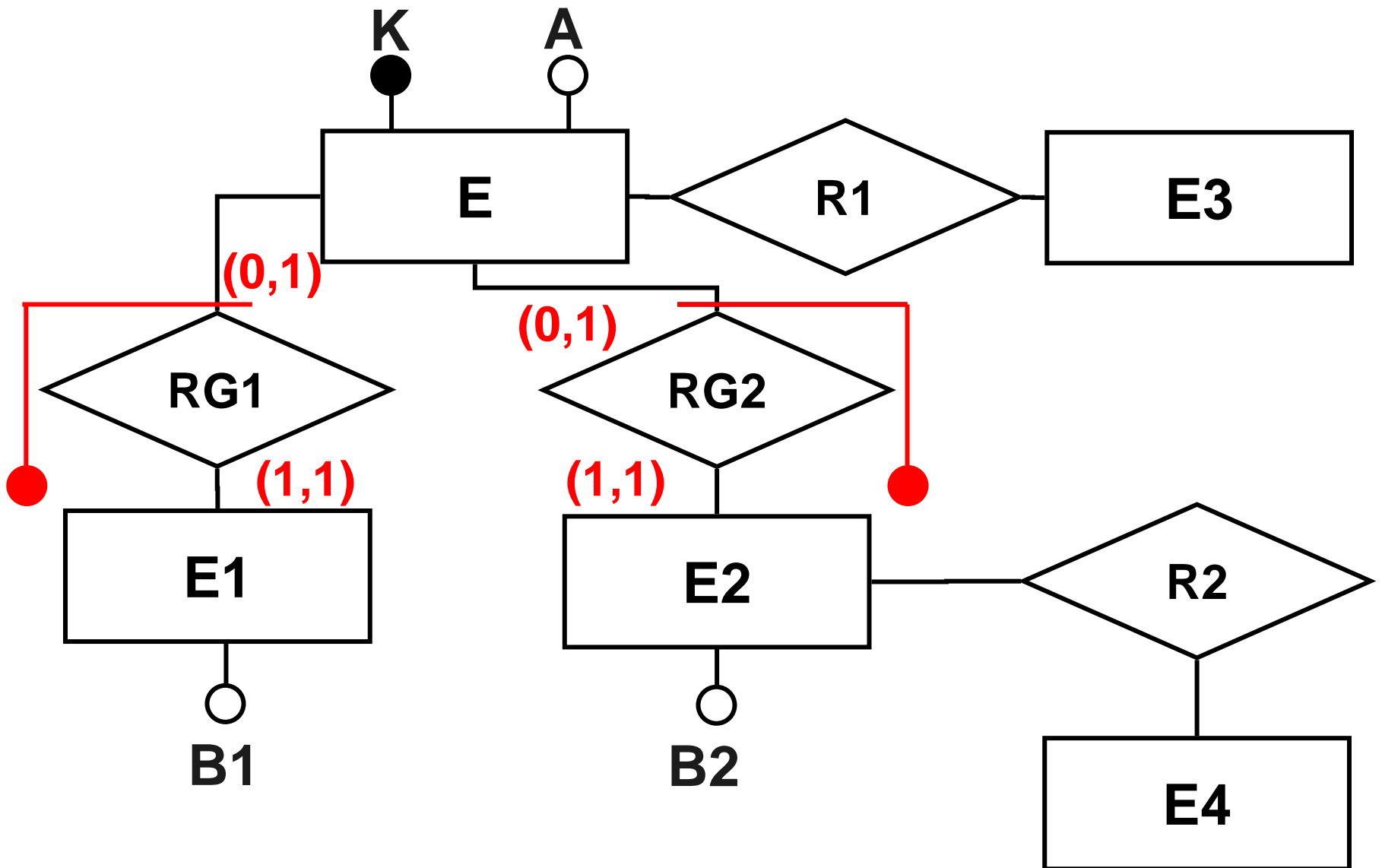
- Se la gerarchia è esclusiva, il vincolo viene perso; nell'esempio questo sarebbe, per la relazione StudIscritti (analogamente per StudErasmus):

```
CONSTRAINT exclusive_hierarchy  
CHECK (NOT EXISTS(SELECT * FROM StudErasmus E WHERE E.Matr = Matr))
```

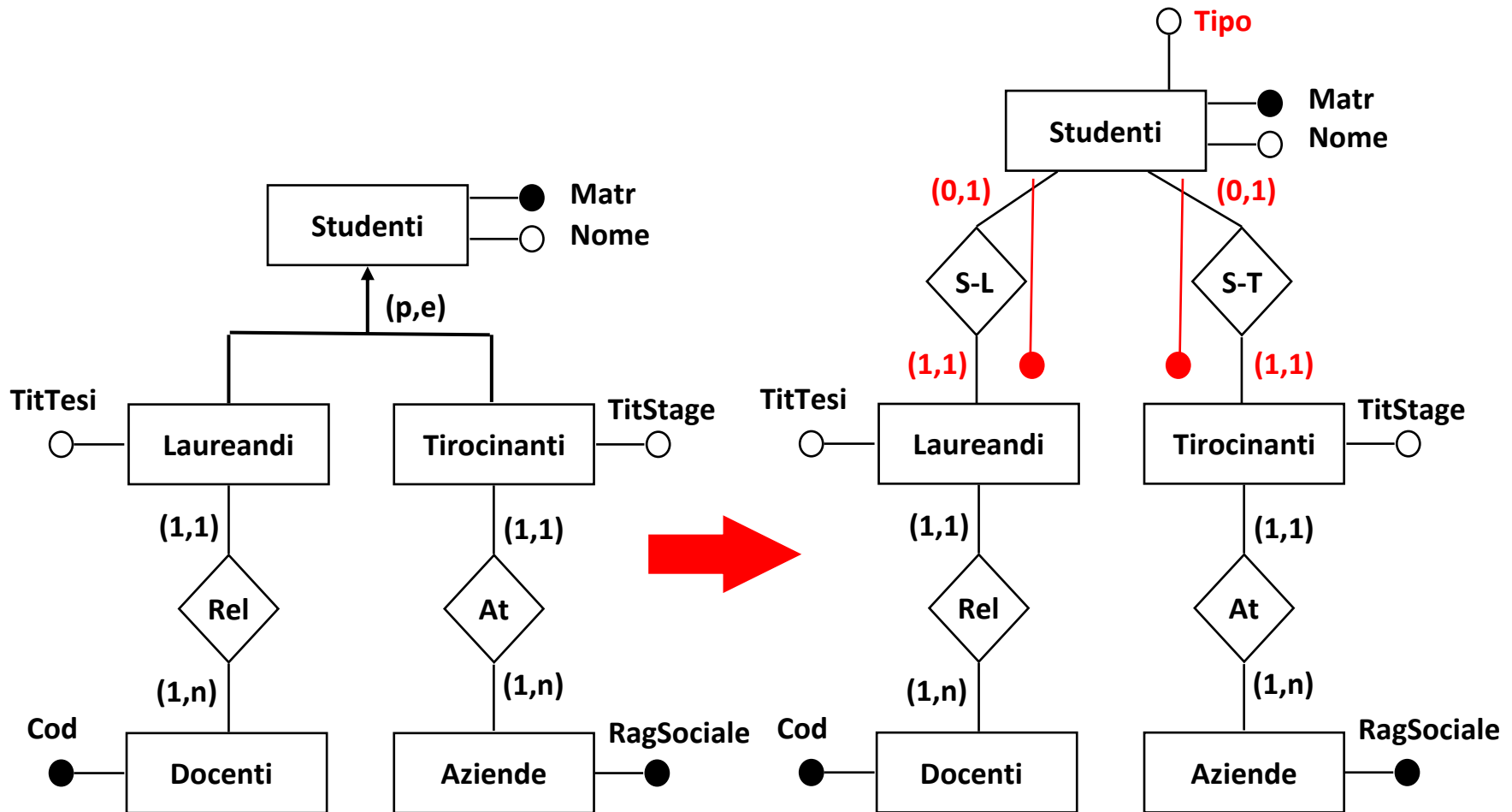
Collasso verso il basso: osservazioni

- Se la copertura non è completa non si può fare
 - non si saprebbe dove mettere le istanze di E che non sono né in E1, né in E2
- Se la copertura non è esclusiva introduce ridondanza
 - una certa istanza può essere sia in E1 che in E2, e quindi si rappresentano due volte i valori degli attributi ereditati da E
- In pratica questa alternativa viene considerata solo nel caso di gerarchie totali ed esclusive
- Per le associazioni che si trasportano sulle entità figlie è necessario garantire che siano ancora verificati i vincoli di cardinalità e/o di foreign key

3. Sostituire con associazioni...



3. Esempio



3. Esempio: traduzione relazionale

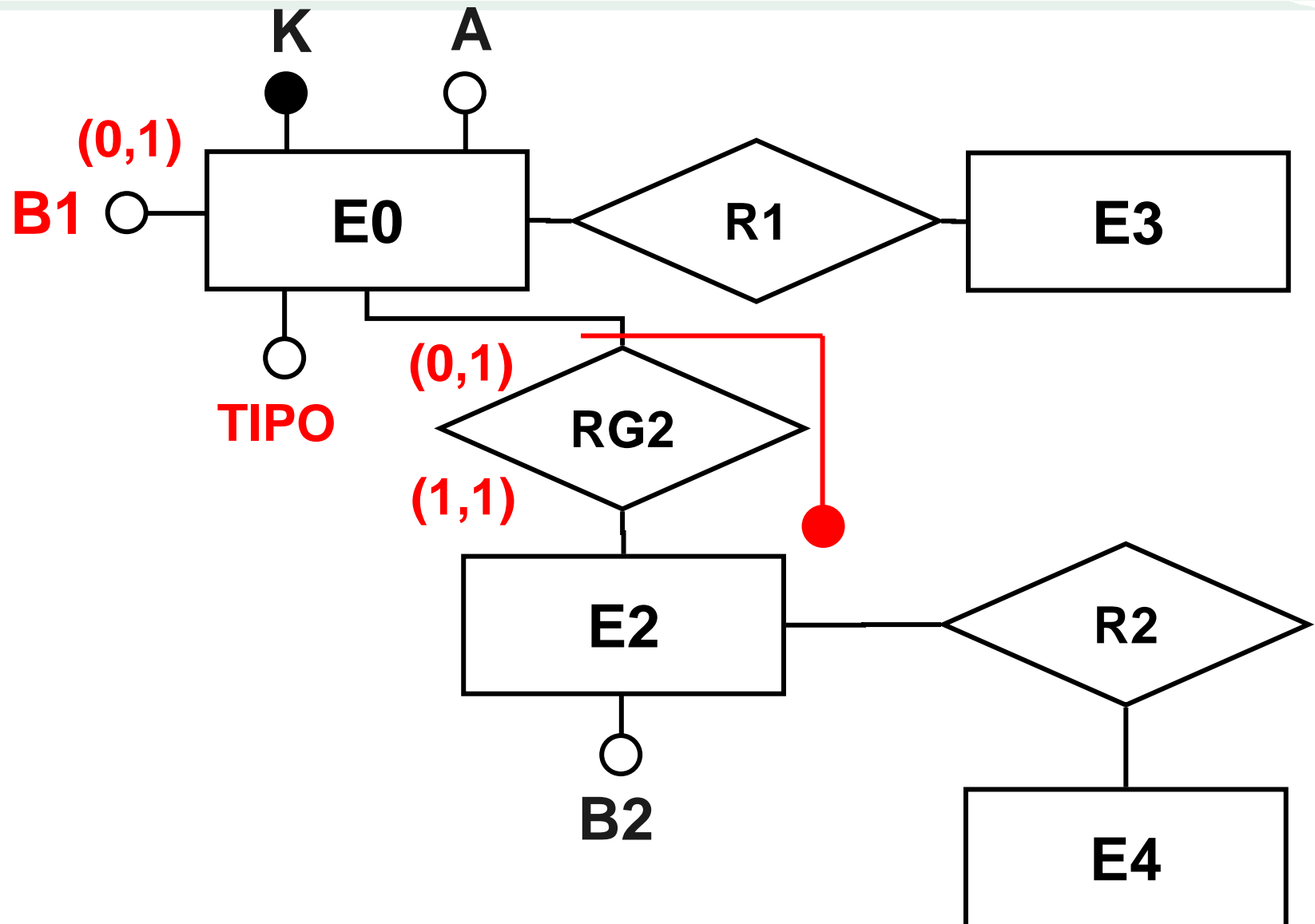
- A livello logico si generano N+1 relazioni:

```
Studenti(Matr, Nome, Tipo)
Laureandi(Matr, TitTesi)
    FK: Matr REFERENCES Studenti
Tirocinanti(Matr, TitStage)
    FK: Matr REFERENCES Studenti
```

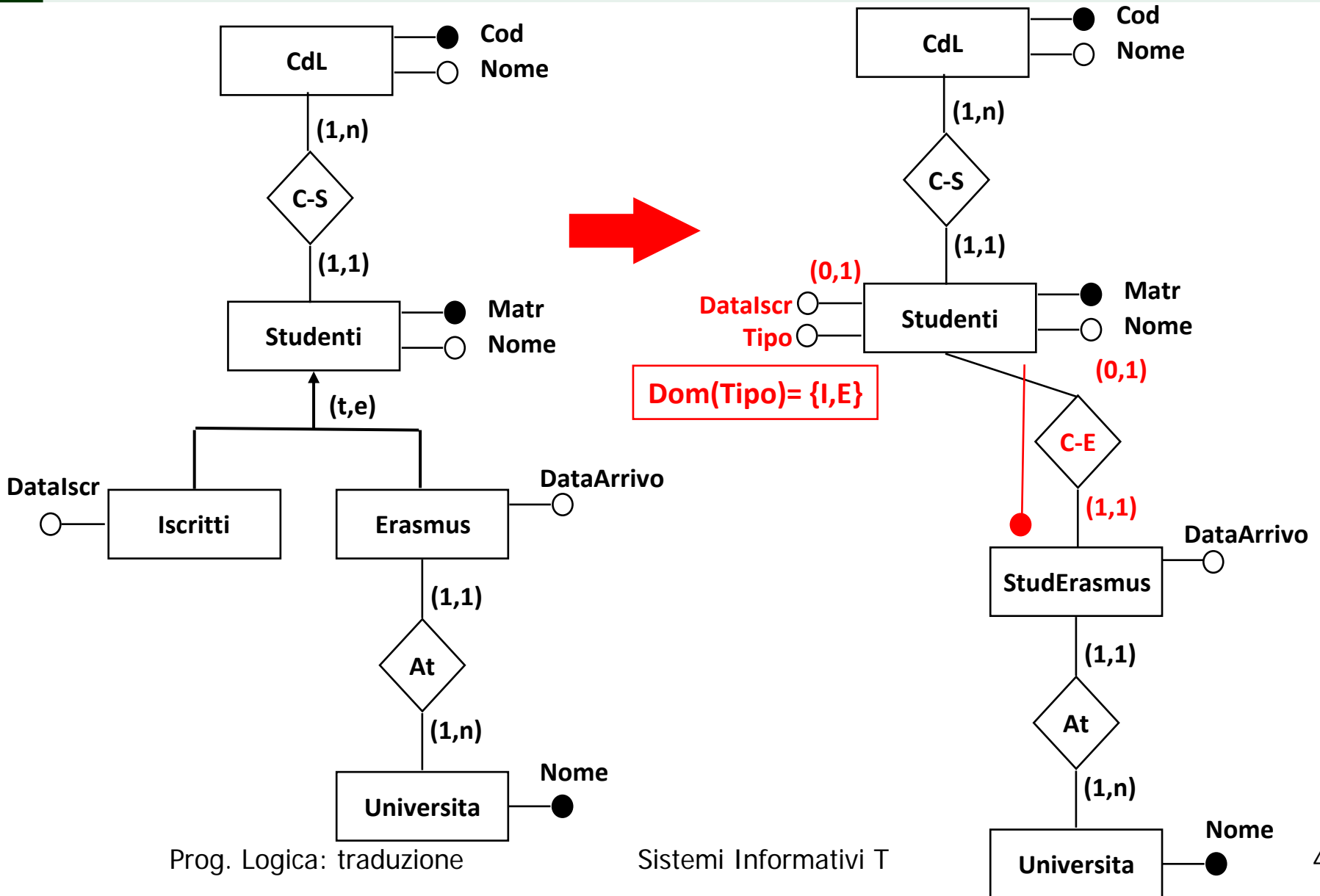
- Vale quanto osservato per il collasso verso il basso: se la gerarchia è esclusiva il vincolo viene perso
 - A tale proposito l'attributo selettore (Tipo) dell'entità genitore può essere utilizzato per aggiungere a Laureandi (analogamente per Tirocinanti) il seguente vincolo:

```
CONSTRAINT ValidMatr
CHECK (EXISTS(SELECT * FROM Studenti S WHERE S.Matr = Matr
AND S.Tipo = 'L'))
```

4. Una soluzione ibrida...



4. Esempio



4. Esempio: traduzione relazionale

```
Studenti(Matr, Nome, Tipo, DataIscriz*)  
  CHECK ((Tipo = 'I' AND DataIscriz IS NOT NULL)  
        OR (Tipo = 'E' AND DataIscriz IS NULL) )  
StudErasmus(Matr, DataArrivo)  
  FK: Matr REFERENCES Studenti
```

- Se si inglobassero le associazioni C-S e At in Studenti e StudErasmus, rispettivamente:

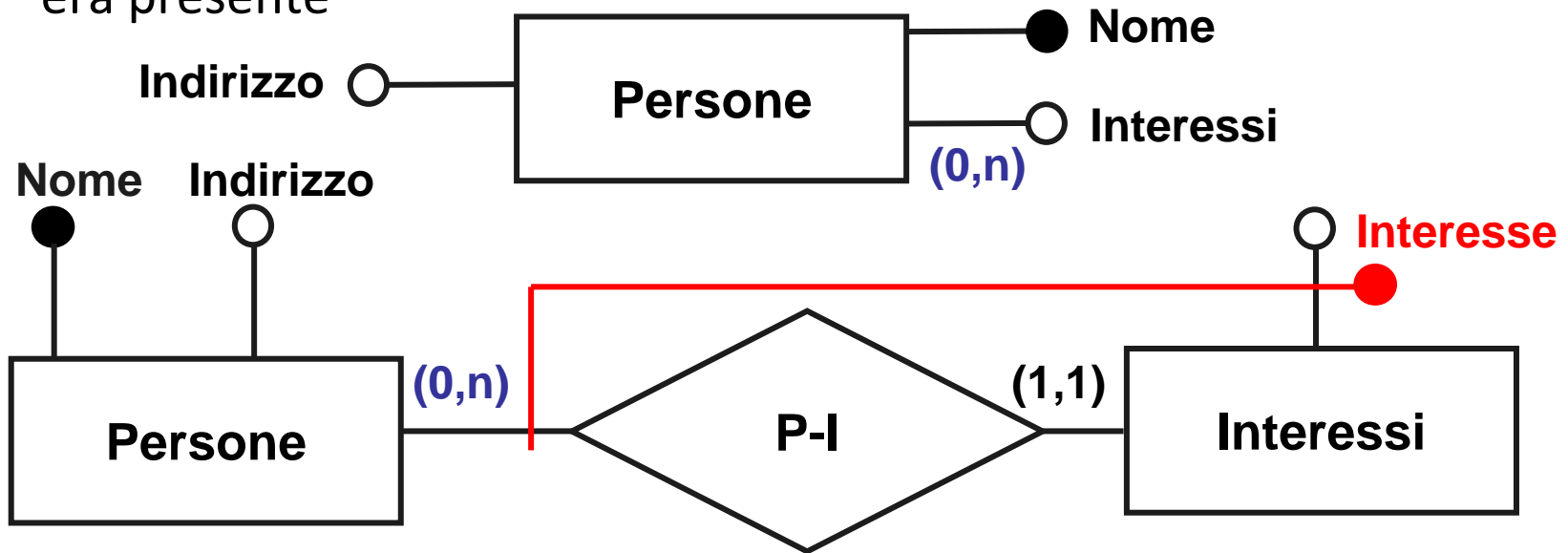
```
Studenti(Matr, Nome, Tipo, DataIscriz*, CodCdL)  
  CHECK ((Tipo = 'I' AND DataIscriz IS NOT NULL)  
        OR (Tipo = 'E' AND DataIscriz IS NULL) )  
  FK: CodCdL REFERENCES CdL  
StudErasmus(Matr, DataArrivo, NomeUniversita)  
  FK: Matr REFERENCES Studenti  
  FK: NomeUniversita REFERENCES Universita
```

Traduzione di attributi multivalore

- Esistono due possibilità, che in entrambi i casi passano per l'introduzione di una **nuova entità**
- Nel primo caso i valori dell'attributo ripetuto possono essere associati a più di un'istanza dell'entità
- Nel secondo ciò non è possibile
 - Si noti che questo vincolo sarebbe stato meglio modellarlo direttamente nello schema E/R

Caso 1: valori condivisi

- Viene introdotta una nuova entità che contiene i valori semplici dell'attributo ripetuto ed è identificata anche dall'entità in cui questi era presente

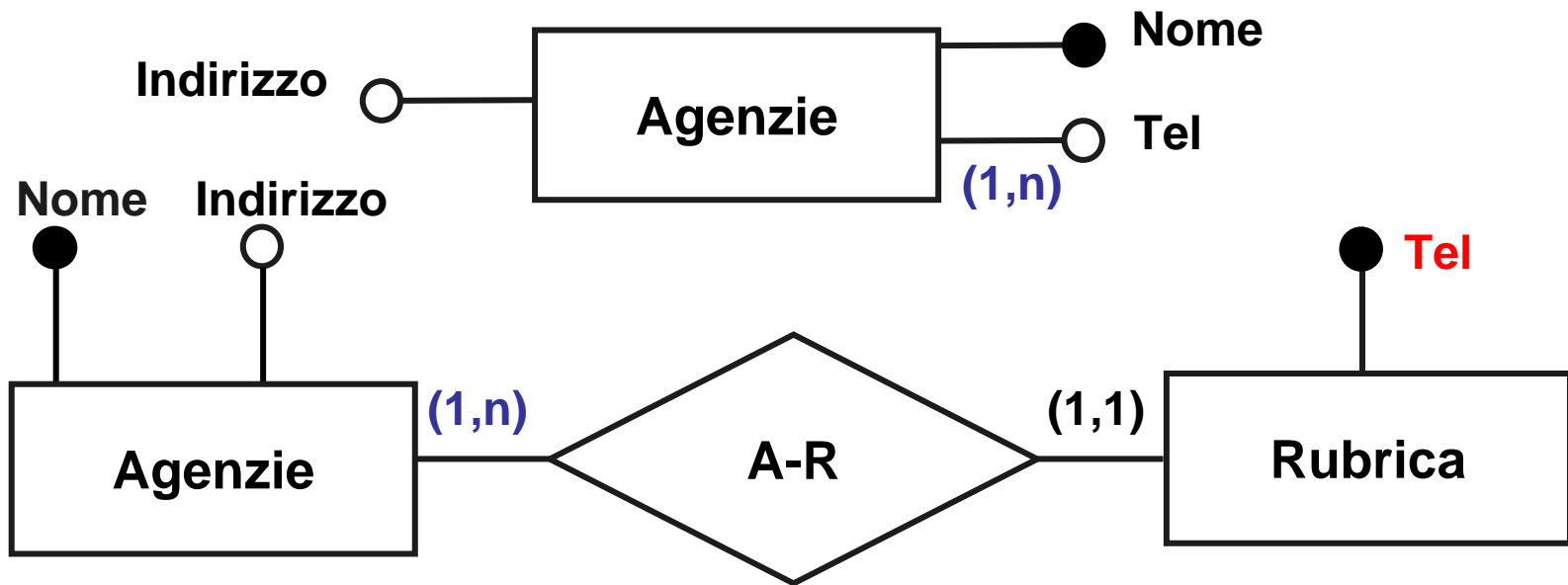


Persone(Nome, Indirizzo)
Interessi(NomePersona, Interesse)
FK: NomePersona REFERENCES Persone

L'identificazione esterna serve a dire che non si vuole una relazione che contenga solo un elenco di interessi!

Caso 2: valori non condivisi

- Viene introdotta una nuova entità che contiene i valori semplici dell'attributo ripetuto ed è identificata solo da questi



```
Agenzie(Nome, Indirizzo)
Rubrica(Tel, NomeAgenzia)
FK: NomeAgenzia REFERENCES Agenzie
```

Si noti che in questo caso non ha senso tradurre l'associazione a sé!

Riassumiamo:

- La **fase di progettazione logica** ha lo scopo di derivare uno schema logico che sia il più fedele possibile (**equivalente**) rispetto allo schema E/R di partenza, e che sia al tempo spesso “efficiente”
- Le diverse alternative che si hanno a disposizione per tradurre le associazioni dipendono dalle **cardinalità massime** in gioco, le quali determinano anche le chiavi delle relazioni che si ottengono
- Le **cardinalità minime** possono portare, in funzione della traduzione scelta, ad avere valori nulli
- Per le **gerarchie** esistono diverse alternative, ognuna delle quali dà luogo a una traduzione che non garantisce l’equivalenza (alcuni vincoli non si esprimono con il DDL di SQL)