Basi di dati

Maurizio Lenzerini

Dipartimento di Informatica e Sistemistica "Antonio Ruberti"
Università di Roma "La Sapienza"

Anno Accademico 2023/2024

Maurizio Lenzerini

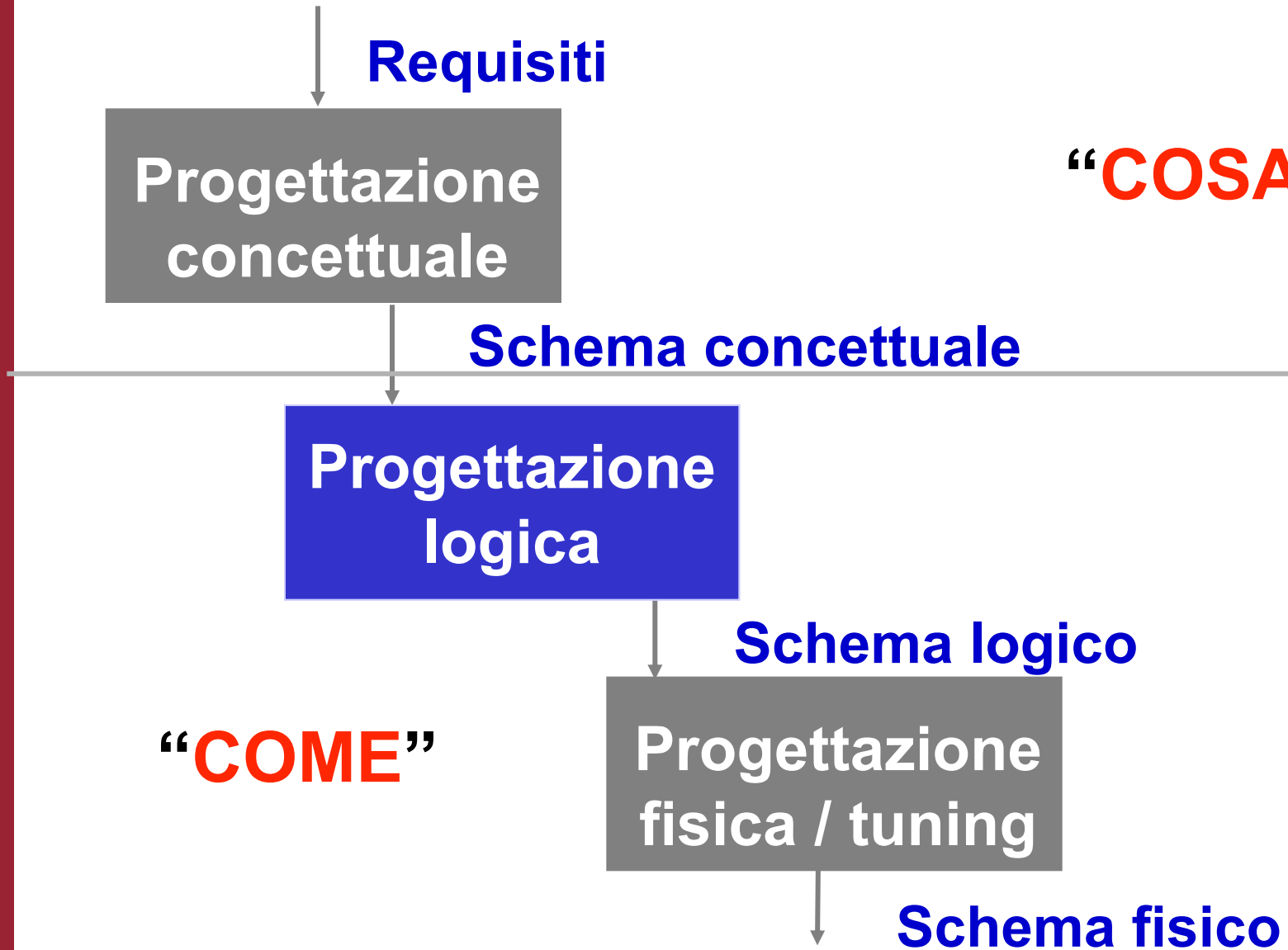
http://www.dis.uniroma1.it/~lenzerin/home/?q=node/44

6. La progettazione logica

6.1 introduzione alla progettazione logica

1. **introduzione alla progettazione logica**
2. ristrutturazione dello schema ER
3. traduzione diretta nel modello relazionale
4. ristrutturazione dello schema logico

Fasi della progettazione



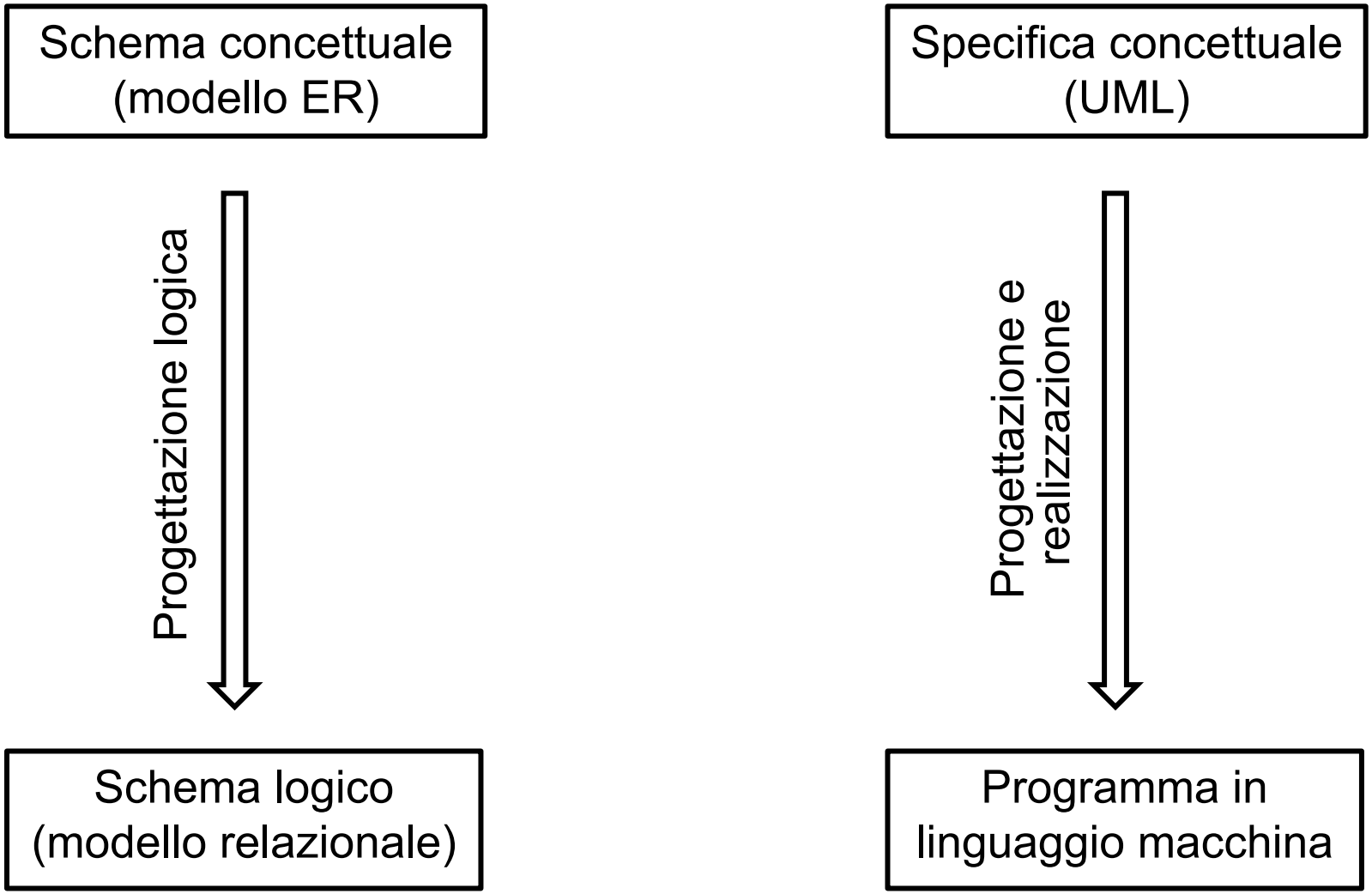
Obiettivo della progettazione logica

Tradurre lo schema concettuale (espresso nel modello ER con vincoli) in uno schema logico che possa rappresentare in modo corretto i dati che codificano il dominio modellato dallo schema concettuale:

- utilizzando il **modello logico** del DBMS scelto
  (nel nostro caso, il modello relazionale con vincoli)

- in maniera **corretta**, **completa** ed **efficiente**

Analogia con la compilazione dei programmi



Dati di ingresso e uscita nella nostra metodologia

Ingresso:

- schema concettuale (diagramma ER e dizionario dei dati)
- informazioni sul carico applicativo

Uscita:

- schema logico espresso nel modello relazionale
- vincoli aggiuntivi
- documentazione associata

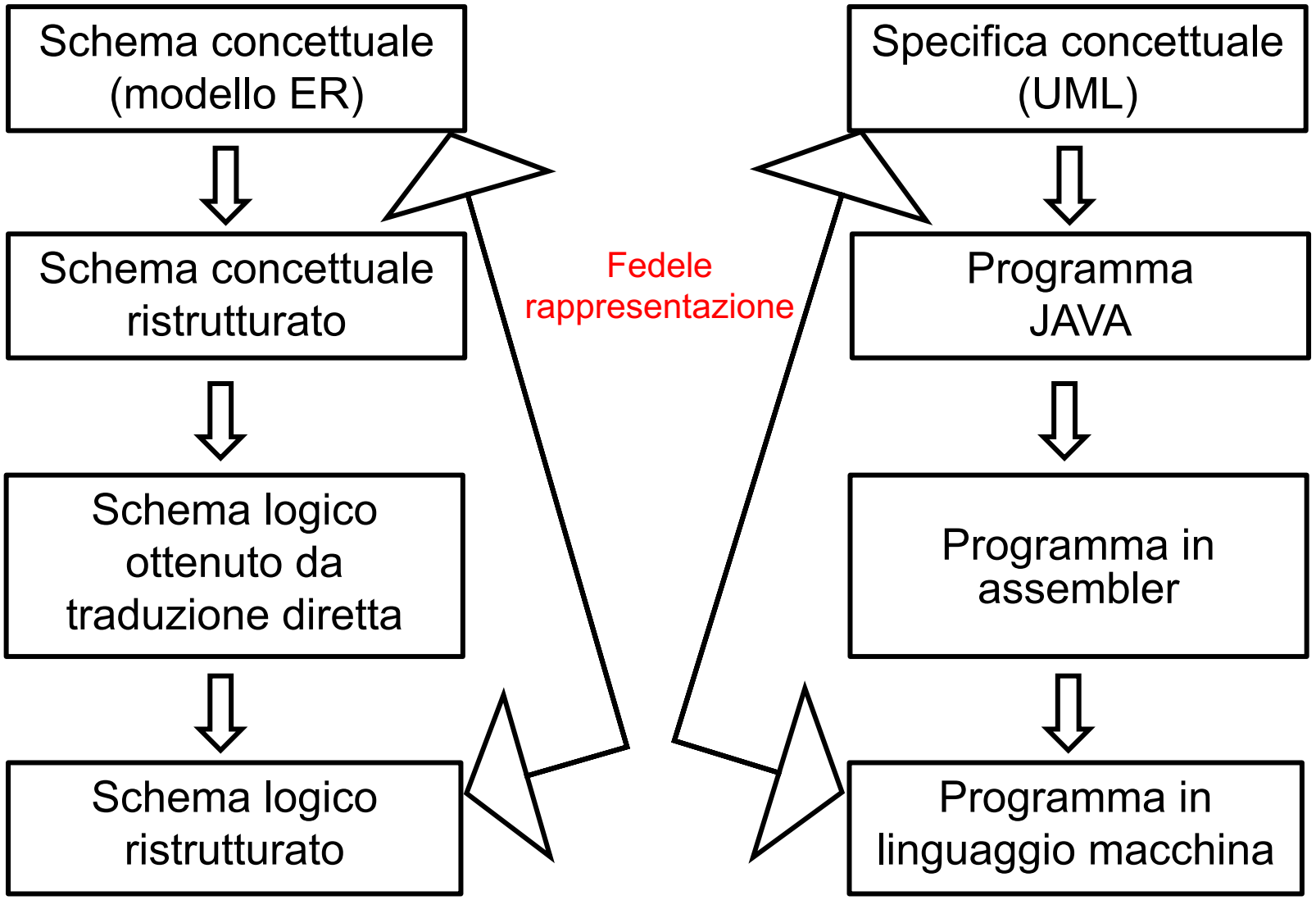
Traduzione guidata dalla necessità di ottimizzare le prestazioni

- La traduzione dello schema concettuale in uno schema relazionale è guidata da due obiettivi: **mantenere la semantica** dello schema concettuale e **ottimizzare le prestazioni**.
- Alcune scelte del processo di traduzione sono fisse e dettate dalla struttura dello schema ER (e quindi dalle scelte effettuate in fase di progettazione concettuale).
- Altre scelte del processo di traduzione sono invece fatte dal progettista con l'esplicito obiettivo di ottimizzare le prestazioni del sistema che verrà realizzato
- Presentiamo una metodologia di progettazione, articolata in diverse fasi, nella quale i momenti in cui il progettista deve effettuare scelte progettuali sono chiaramente delimitati.

Fasi della progettazione logica

1. **Ristrutturazione dello schema ER**:
   - eliminazione (mantenendone la semantica) dei costrutti non direttamente traducibili nel modello relazionale
   - scelta degli identificatori principali delle entità e delle relazioni
2. **Traduzione diretta** dello schema ER ristrutturato nel modello relazionale:
   - la traduzione è **diretta**, nel senso che può essere eseguita automaticamente o quasi automaticamente, non richiedendo scelte da parte del progettista, se non in situazioni eccezionali
   - il risultato della traduzione diretta tiene conto delle scelte fatte in fase di progettazione concettuale
   - lo schema relazionale prodotto **non** contiene ridondanze, se non quelle volute
3. **Ristrutturazione dello schema relazionale**:
   - richiede delle scelte da parte del progettista, tenendo conto dell'ottimizzazione delle prestazioni (carico applicativo)

Analogia con la compilazione dei programmi



6. La progettazione logica

6.2 ristrutturazione dello schema ER

1. introduzione alla progettazione logica
2. **ristrutturazione dello schema ER**
3. traduzione diretta nel modello relazionale
4. ristrutturazione dello schema logico

Ristrutturazione dello schema ER

Motivazioni:

- **semplificare** la successiva fase di traduzione nel modello relazionale eliminando quei costrutti non direttamente traducibili (mantenendone però la semantica)

- tenere conto di aspetti relativi all'**efficienza**

Osservazione:

- uno schema ER **ristrutturato** è uno schema ER **degradato** opportunamente (cioè "monco" di alcuni costrutti) allo scopo di avvicinarsi al modello relazionale

Attività della ristrutturazione dello schema ER

0. preparazione iniziale dello schema ER
1. analisi delle ridondanze
2. eliminazione degli attributi multivalore
3. eliminazione degli attributi composti
4. eliminazione delle ISA e delle generalizzazioni
5. scelta degli identificatori principali di entità e relazioni
6. specifica degli ulteriori vincoli esterni

Ristrutturazione – attività 0: preparazione iniziale dello schema ER

- Lo scopo di questa fase è fare in modo che il progettista abbia il quadro completo di quanto rappresentato nello schema, in modo che nessun aspetto dello schema concettuale venga sottovalutato nella fase di progettazione logico.
- In questa fase viene quindi ridisegnato il diagramma dello schema ER aggiungendo al diagramma tutti i vincoli (se ce ne sono) che sono considerati rilevanti ma che, essendo implicati da quelli presenti nello schema, non erano stati indicati esplicitamente nello schema concettuale.
- In particolare, si suggerisce la seguente strategia:
  - Per ogni entità E nello schema ER, i vincoli di identificazione rilevanti (anche quelli implicati da altri vincoli) per l'entità E vengono riportati nello schema ER.
  - Per ogni relazione R nello schema ER, tutti i vincoli di identificazione rilevanti (anche quelli derivati da altri) per la relazione R, vengono riportati nello schema ER. Tipicamente, quello implicito (ovvero formato da tutti i ruoli della relazione) non viene riportato.

Quelli appena descritti sono suggerimenti. In realtà, è importante osservare che secondo la nostra metodologia, nello schema ristrutturato non è necessario riportare tutti i vincoli derivati. È necessario però riportare tutti quelli che si reputano rilevanti, ovvero che non si vuole rischiare di ignorare per errore.

Ristrutturazione – attività 1: analisi delle ridondanze

- Se le ridondanze intensionali dovrebbero essere evitate, una **ridondanza** estensionale in uno schema ER prefigura la presenza nella base di dati di dati significativi, ma derivabili da altri.
- Sappiamo che le ridondanze estensionali, se presenti, devono documentate nello schema ER (ovvero espresse attraverso **vincoli**).
- In questa fase si decide se eliminare le ridondanze eventualmente presenti o mantenerle, e se introdurne delle nuove.
- **Vantaggi** nel mantenere una ridondanza:
  - potenziale maggiore efficienza nella esecuzione delle interrogazioni
- **Svantaggi** nel mantenere una ridondanza:
  - gestione dei vincoli aggiuntivi
  - appesantimento degli aggiornamenti
  - maggiore occupazione di spazio

Analisi delle ridondanze

Abbiamo visto che le forme più comuni di ridondanza estensionale in uno schema ER sono causate da:

- attributi derivabili:
  - da altri attributi della stessa entità (o relazione)
  - da attributi di altre entità (o relazioni)
- relazioni derivabili dalla composizione di altre relazioni in presenza di cicli

Per ciascuna ridondanza bisogna valutare, in funzione del carico applicativo previsto (aggiornamenti, interrogazioni, occupazione di spazio) se è opportuno mantenerla oppure eliminarla. Se si sceglie di mantenere la ridondanza, questa deve essere **documentata** (attraverso opportuni vincoli).

Nel seguito, se non specificato altrimenti, assumeremo di eliminare ogni forma di ridondanza.

Ristrutturazione – attività 2: eliminazione di attributi multivalore

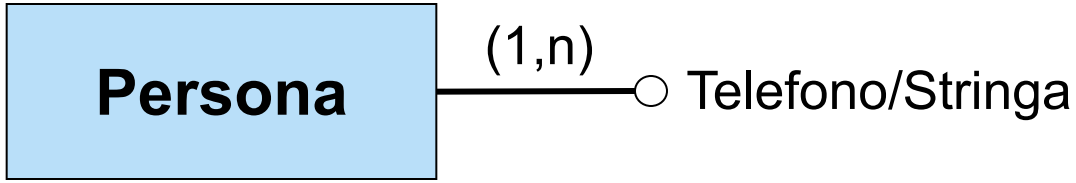
- In generale, un **attributo multivalore** (ovvero un attributo con cardinalità massima maggiore di 1) non può essere tradotto direttamente nel modello relazionale senza introdurre delle ridondanze inaccettabili nelle relazioni ottenute.
- Dobbiamo quindi eliminare tutti gli attributi multivalore.
- L'eliminazione di un **attributo multivalore di un'entità** si effettua trasformando l'attributo in una relazione binaria, ed introducendo un'opportuna entità per il dominio (cfr. la parte sulla progettazione concettuale sulle trasformazioni).
- L'eliminazione di un **attributo multivalore di una relazione** può essere realizzata mediante la preventiva trasformazione della relazione in un'entità (cfr. la parte sulla progettazione concettuale sulle trasformazioni).

Nota: se trasformiamo una relazione R in un'entità, ed R era in ISA o in una gerarchia con altre relazioni, allora anche queste devono essere trasformate in entità (cfr. la parte sulla progettazione concettuale sulle trasformazioni).

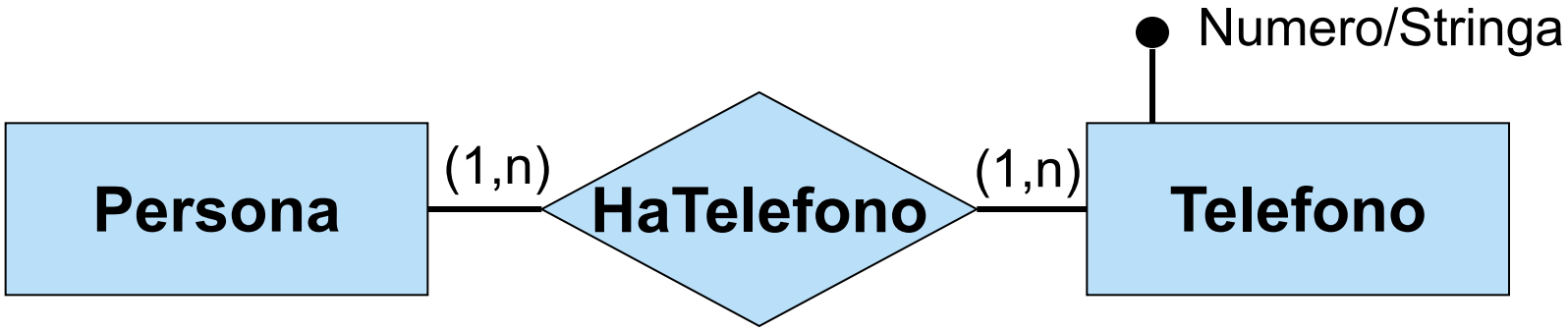
Eliminazione di attributi multivalore di entità

Si trasforma l'attributo multivalore dell'entità in una relazione e il corrispondente dominio in entità.

Esempio:



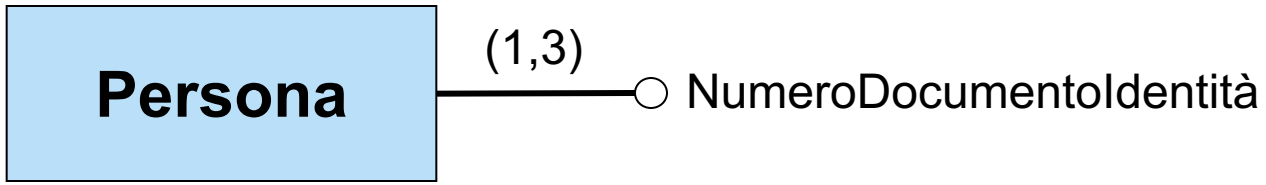
Si trasforma in:



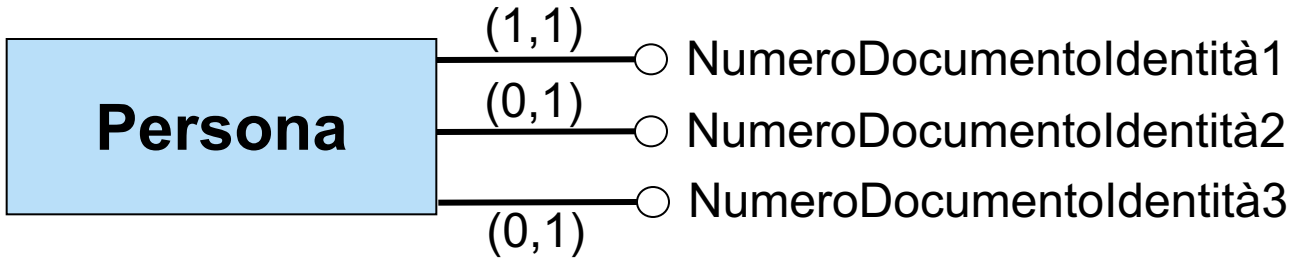
Eliminazione di attributi multivalore di entità

In linea di principio, nel caso particolare di cardinalità massima M uguale ad una costante (e quindi diversa da *n*) maggiore di 1, si può trasformare l'attributo multivalore in M attributi.

Esempio con cardinalità minima 1 e massima 3:



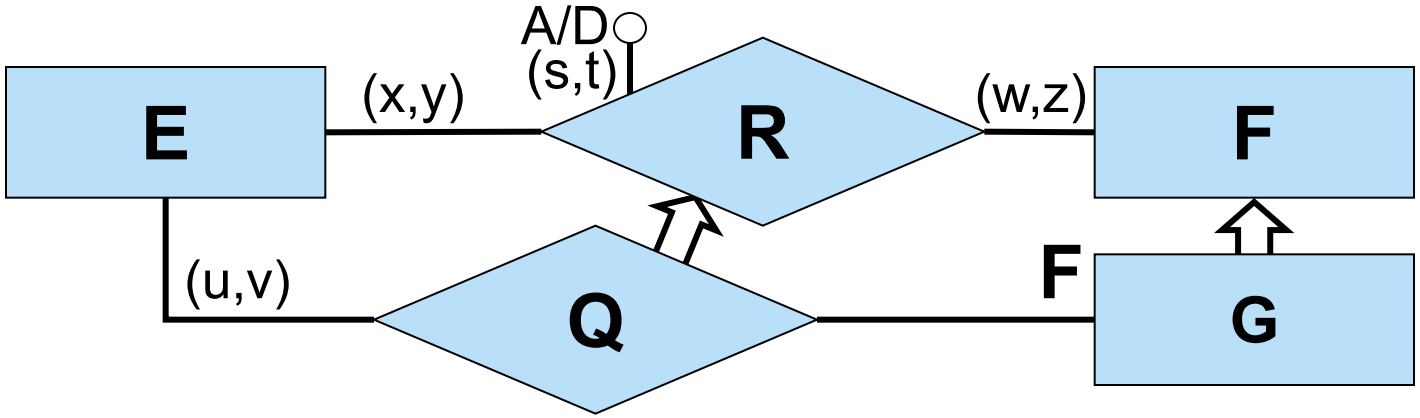
Si può trasformare in:



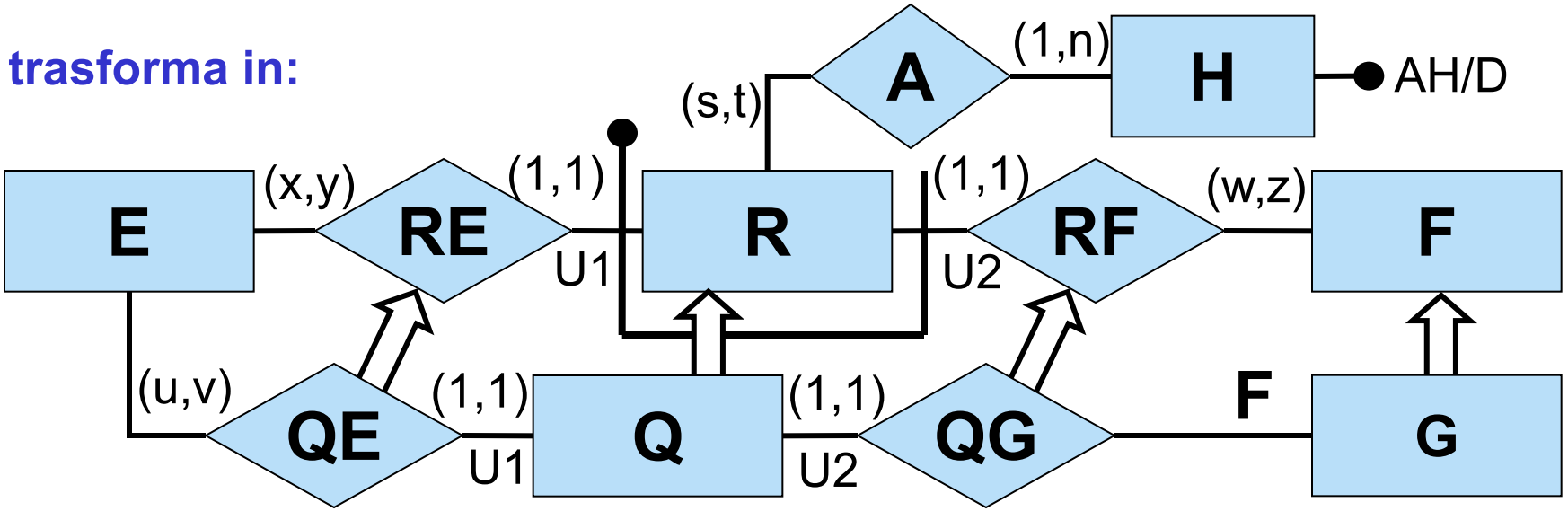
Tuttavia questa soluzione non è comunemente adottata perché da una parte introduce nuovi attributi e dall'altra può causare l'introduzione di diversi valori nulli nella base di dati.

Eliminazione di attributi multivalore di relazioni

Qualunque sia la cardinalità (s,t) dell'attributo si può trasformare la relazione R in entità e l'attributo multivalore di R in una relazione. Anche eventuali relazioni in ISA con R devono essere trasformate in entità.

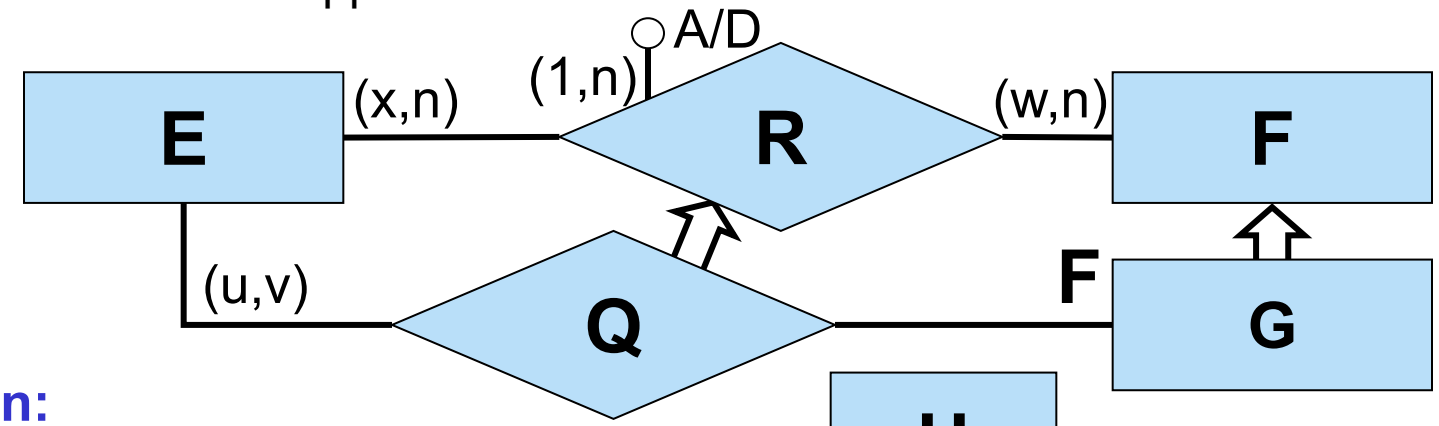


Si trasforma in:

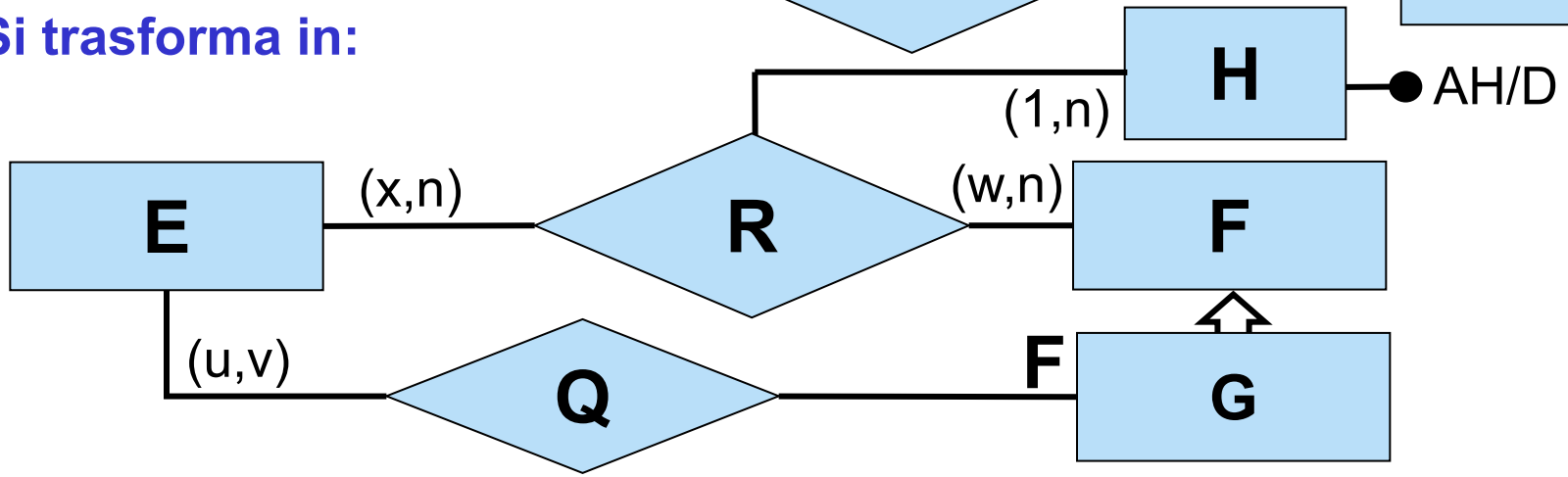


Eliminazione di attributi multivalore di relazioni

Nel caso particolare in cui la cardinalità dell'attributo multivalore sia (1,n) e le cardinalità massime di tutti i ruoli di R sia n, si può effettuare una diversa trasformazione: si trasforma l'attributo multivalore di R in un'entità che viene connessa alla relazione R (che quindi aumenta la arietà). Eventuali relazioni in ISA con R devono essere trattate opportunamente.



Si trasforma in:



Vincolo esterno: per ogni <E:e,F:f> in Istanze(I,Q) esiste almeno un h in Istanze(I.H) tale che <E:e,F:f,H:h> è in Istanze(I,R).

Ristrutturazione – attività 3: eliminazione di attributi composti

Un **attributo composto** di un'entità (o di una relazione) a questo punto ha cardinalità (1,1) oppure (0,1).

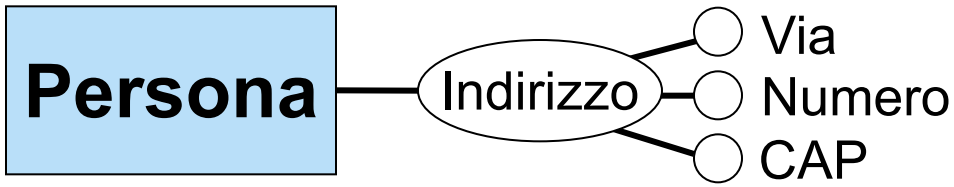
- Se l'attributo ha cardinalità (1,1), si associano direttamente gli attributi componenti all'entità (o alla relazione).
- Se l'attributo ha cardinalità (0,1) si può
  - procedere come per gli attributi (1,1), ma con l'avvertenza che rappresentare l'opzionalità richiede un vincolo esterno
  - oppure trasformare l'attributo composto in una relazione binaria introducendo una nuova entità, come fatto per gli attributi multivalore (cfr. parte 4, pagina 218), mantenendo la cardinalità (0,1) sulla relazione.

    Ovviamente, se l'attributo composto è di una relazione, è necessario trasformare tale relazione in un'entità.

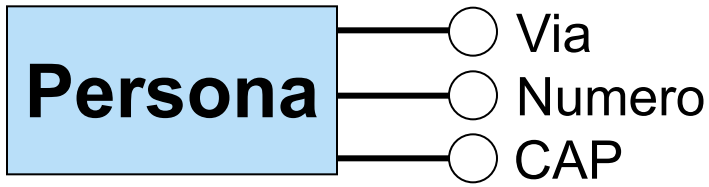
Eliminazione di attributi composti: alternativa 1

Si associano direttamente gli attributi componenti all'entità (o alla relazione) a cui è associato l'attributo:

Esempio:

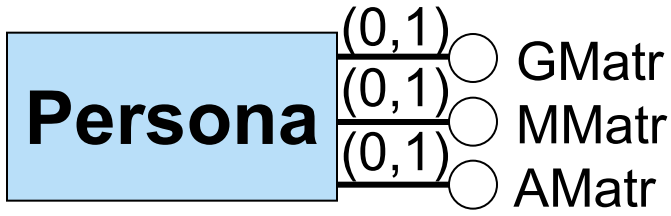


viene trasformato in:



Esempio:

viene trasformato in:

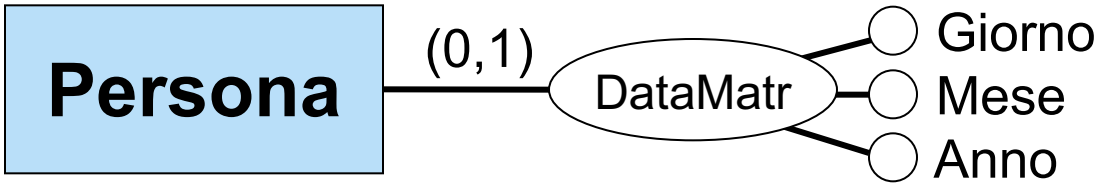


... con il vincolo esterno: per ogni istanza di Persona, ciascun attributo tra GMatr, MMatr e AMatr è definito se e solo se lo sono anche gli altri due.

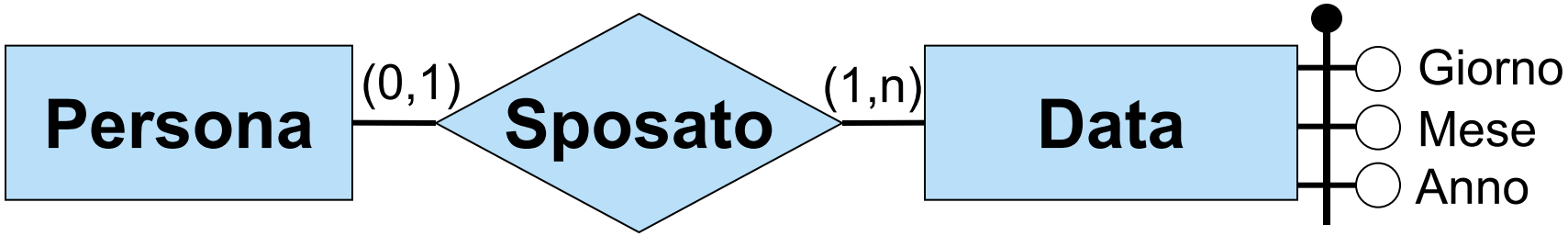
Eliminazione di attributi composti: alternativa 2

Si trasforma l'attributo composto in una relazione binaria e si introduce una nuova entità.

Esempio:



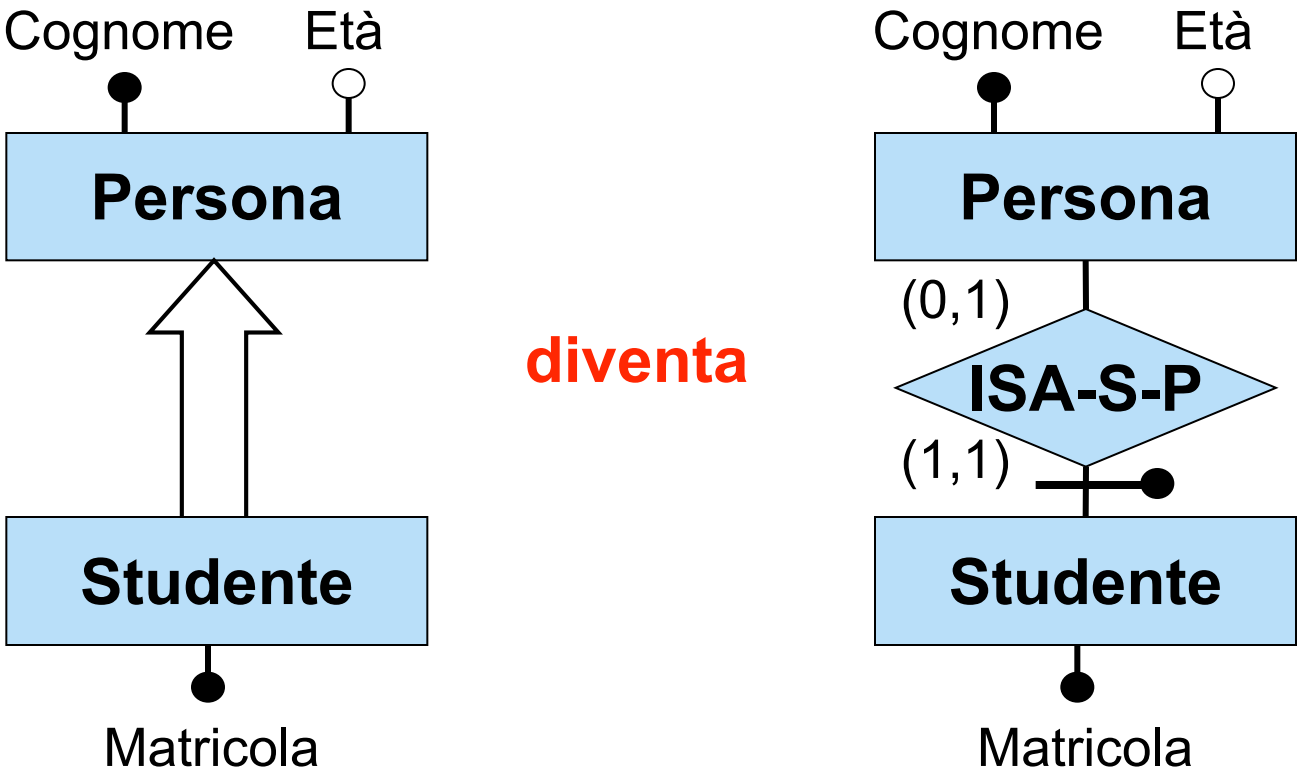
Viene trasformato in:



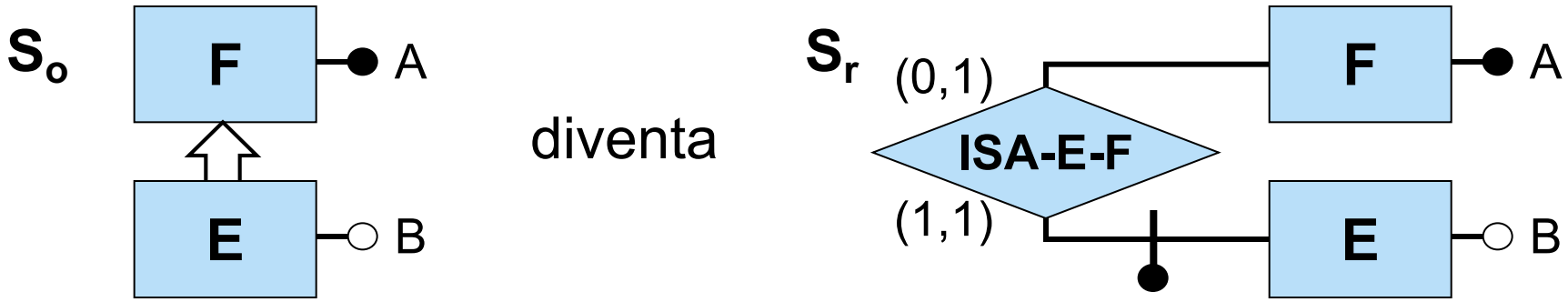
Ristrutturazione – attività 4: eliminazione di ISA tra entità

Una relazione **E ISA F** tra due entità **E** ed **F** viene sostituita da una nuova relazione binaria **ISA-E-F** tra **E** ed **F** a cui **E** partecipa con cardinalità (1,1) e **F** con cardinalità (0,1). Agli eventuali identificatori di **E** viene aggiunto un identificatore esterno dato dalla partecipazione ad **ISA-E-F**.

Esempio:



Eliminazione di ISA tra entità: livello estensionale



Istanza I di So:
istanze(I,F) = { f_1, f_2, f_3 }
istanze(I,E) = { f_1, f_2 }
istanze(I,A) = {(f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(I,B) = { (f_1,b_1), (f_2 ,b_2) }

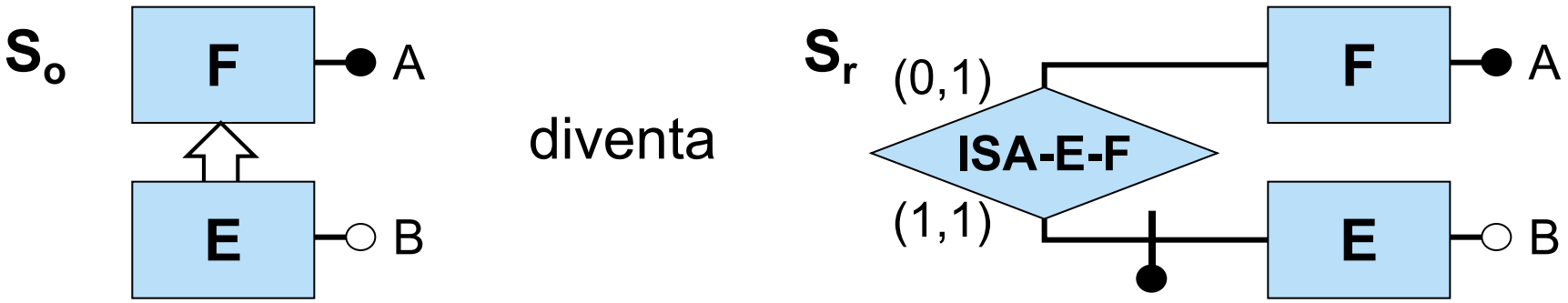
Istanza J di Sr:
istanze(J,F) = { f_1, f_2, f_3 }
istanze(J,E) = { e_1, e_2 }
istanze(J,A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3) }
istanze(J,B) = { (e_1,b_1), (e_2 ,b_2) }
istanze(J,ISA-E-F) = {(E:e_1,F:f_1),
(E:e_2,F:f_2)}

Si noti che nello schema S_r risultante dall'eliminazione delle ISA da S_o, **tutte le entità sono disgiunte a coppie**, e quindi non hanno più istanze in comune (su questo torneremo fra breve).

Eliminazione di ISA tra entità: corrispondenza tra istanze

Esiste una stretta corrispondenza tra le istanze di uno schema S_o e le istanze dello schema S_r ottenuto da S_o eliminando le ISA tra entità.

Mostriamo questa proprietà tramite un esempio:

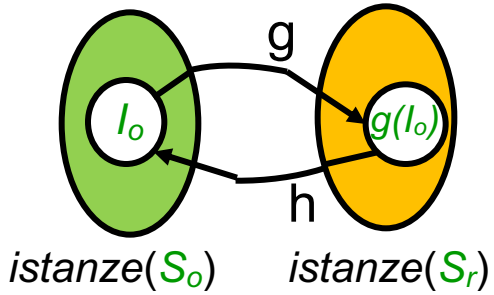


In particolare, mostriamo che esistono due funzioni g ed h tali che:

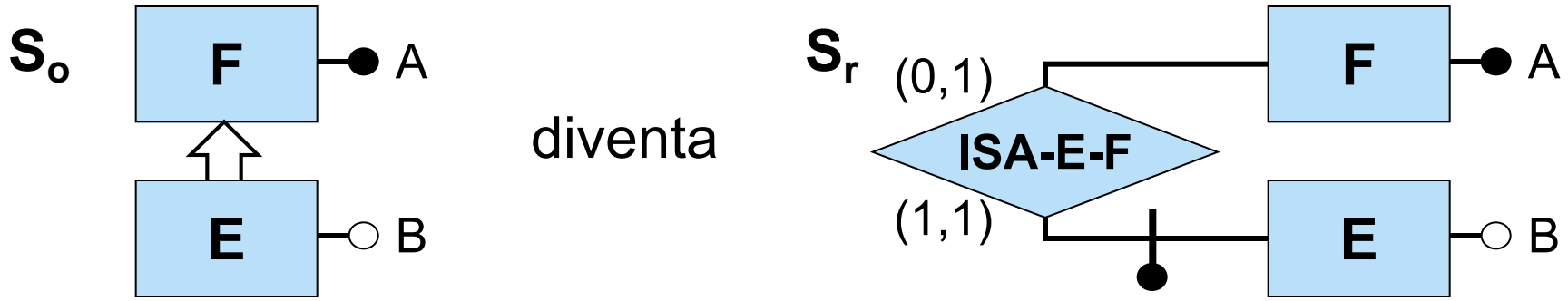
– g è una funzione totale da *istanze*(S_o) a *istanze*(S_r)

– h è una funzione totale da *istanze*(S_r) a *istanze*(S_o)

– per ogni istanza I_o di S_o, si ha che $h(g(I_o)) = I_o$



Corrispondenza tra istanze – funzione g



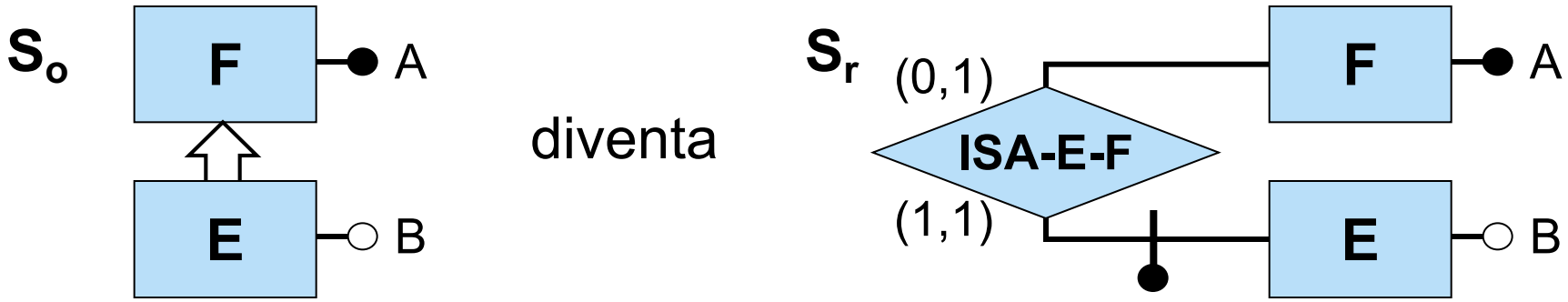
Definiamo la funzione g: istanze(S_o) → istanze(S_r) in modo che, ad un'istanza I_o di S_o, g assegni un'istanza I_r di S_r definita nel seguente modo:

- *istanze*(I_r,F) = *istanze*(I_o,F)
- *istanze*(I_r,A) = *istanze*(I_o,A)
- per definire *istanze*(I_r,E), introduciamo in I_r, per ogni $x \in$ *istanze*(I_o,E), un nuovo oggetto $g_E(x)$, e definiamo

  istanze(I_r,E) = { $g_E(x)$ | $x \in$ *istanze*(I_o,E) }
- *istanze*(I_r,ISA-E-F) = { (E:$g_E(x)$,F:x) | $x \in$ *istanze*(I_o,E) }
- *istanze*(I_r,B) = { ($g_E(x)$,b) | (x,b) $\in$ *istanze*(I_o,B) }

È facile verificare che I_r così definita è effettivamente un'istanza di S_r.

Corrispondenza tra istanze – funzione h

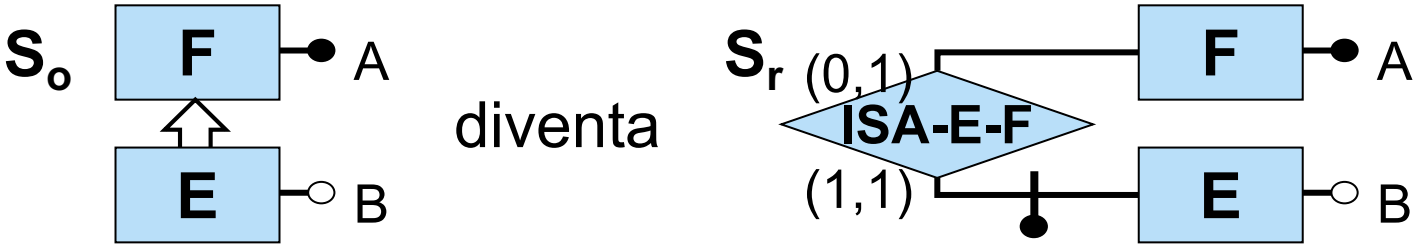


Definiamo la funzione h: istanze(S_r) → istanze(S_o) in modo che, ad un'istanza I_r di S_r, h assegni l'istanza I_o di S_o definita nel seguente modo:

- *istanze*(I_o,F) = *istanze*(I_r,F)
- *istanze*(I_o,A) = *istanze*(I_r,A)
- *istanze*(I_o,E) = { x∈*istanze*(I_r,F) | esiste y∈*istanze*(I_r,E) tale che (E:y,F:x)∈*istanze*(I_r,ISA-E-F) }
- *istanze*(I_o,B) = { (x,b) | x∈*istanze*(I_o,E), esiste y∈*istanze*(I_r,E) tale che (E:y,F:x)∈*istanze*(I_r,ISA-E-F) e (y,b)∈*istanze*(I_r,B) }

È facile verificare che I_o così definita è effettivamente un'istanza di S_o, e che inoltre h(g(I_o)) = I_o.

Osservazione sullo schema risultante



Istanza I di So:
istanze(I,F) = { f_1, f_2, f_3 }
istanze(I,E) = { f_1, f_2 }
istanze(I,A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(I,B) = { (f_1,b_1), (f_2 ,b_2) }

Istanza J di Sr:
istanze(J,F) = { f_1, f_2, f_3 }
istanze(J,E) = { e_1, e_2 }
istanze(J,A) = { (f_1,a_1), (f_2 ,a_2), (f_3,a_3)}
istanze(J,B) = { (e_1,b_1), (e_2 ,b_2) }
istanze(J,ISA-E-F) = {(E:e_1,F:f_1), (E:e_2,F:f_2) }

Come osservato prima, nello schema S_r risultante dall'eliminazione delle ISA da S_o, **tutte le entità sono disgiunte a coppie**.

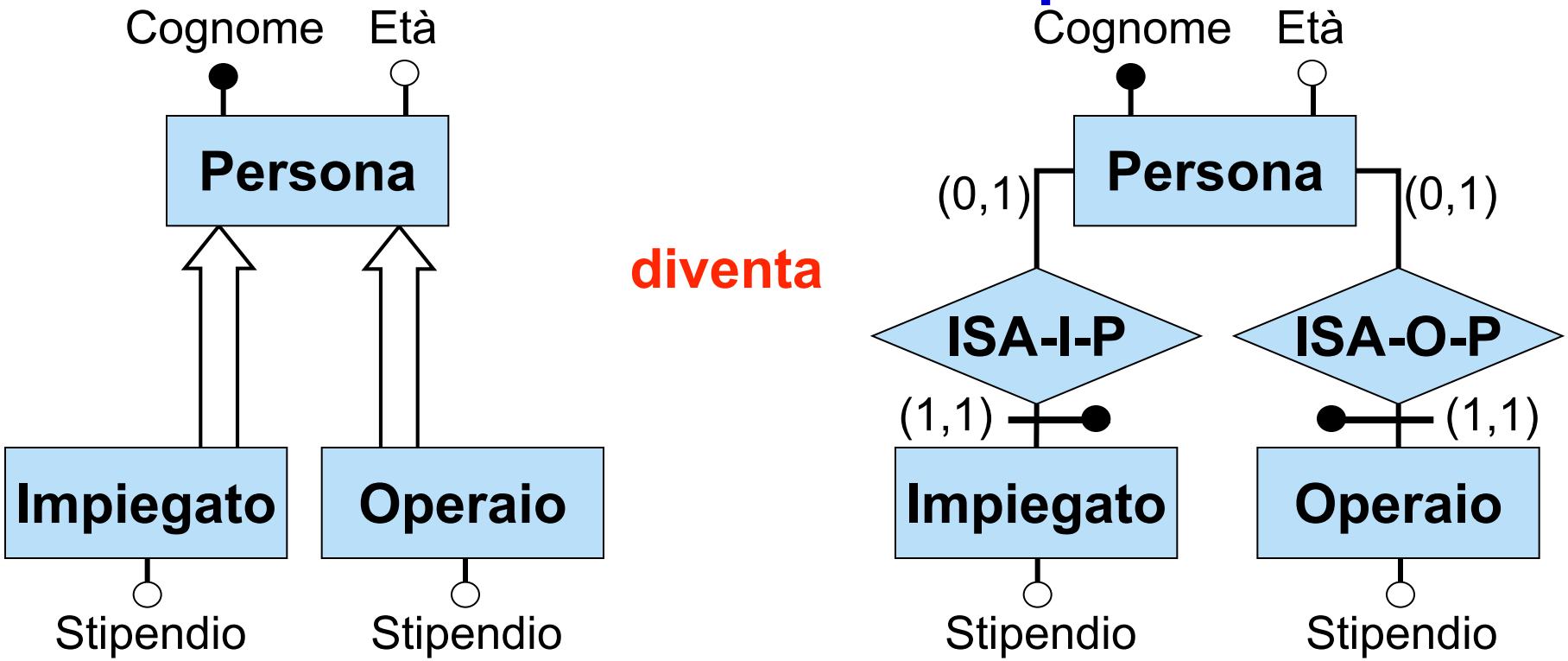
Come si concilia questa osservazione con il fatto che nello schema originario ciò non era vero? La risposta sta nelle funzioni g ed h illustrate precedentemente. Tramite queste funzioni è possibile infatti stabilire se due qualunque istanze di entità nello schema risultante corrispondono ad un'unica istanza di entità nello schema originario (ad es., e_1 ed f_1 nella istanza di S_r corrispondono entrambi ad f_1 nella istanza di S_o).

Attributi in comune nella eliminazione di ISA tra entità

Se due entità non disgiunte nello schema originario hanno un attributo A in comune, applicando la trasformazione per eliminare la relazione ISA, nelle due corrispondenti entità disgiunte dello schema risultante troviamo due attributi di nome A, che di fatto rappresentano due funzioni (o relazioni, se A è multivalore) diverse.

Al fine di mantenere la semantica originaria, dobbiamo quindi imporre che le due funzioni, quando applicate a due oggetti e, f dello schema risultante che corrispondono allo stesso oggetto nello schema di partenza, assegnino ad e ed f lo stesso valore. Ciò viene fatto con un vincolo esterno sullo schema ristrutturato.

Attributi in comune nella eliminazione di ISA tra entità: esempio



con il **vincolo esterno** nello schema risultante:

per ogni istanza I dello schema, per ogni $p \in$ *istanze*(I,Persona), se esistono $i \in$ *istanze*(I,Impiegato), e $o \in$ *istanze*(I,Operaio) tali che $(i,p) \in$ *istanze*(I,ISA-I-P) e $(o,p) \in$ *istanze*(I,ISA-O-P), allora esiste s tale che <i,s> $\in$ Istanze(I,Stipendio) e <o,s> $\in$ Istanze(I,Stipendio)

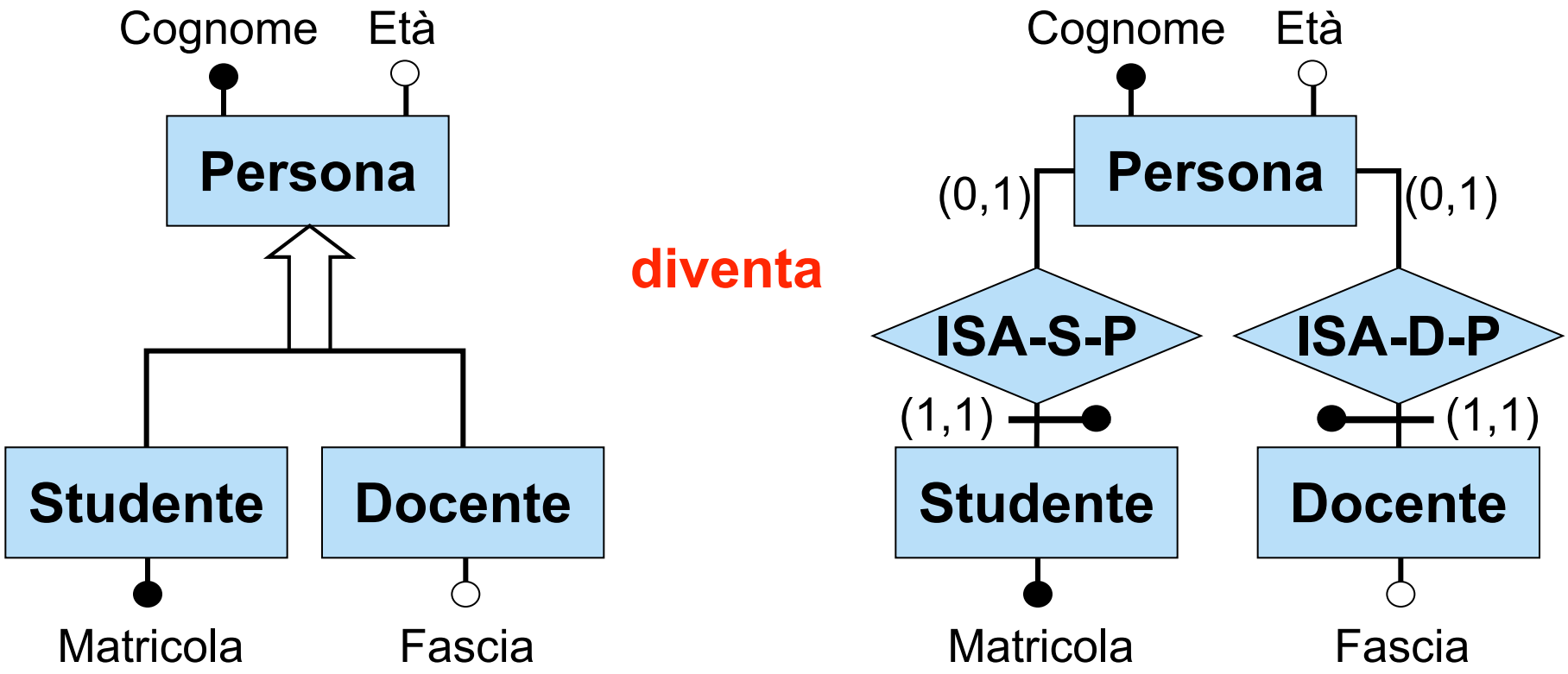
Ristrutturazione – attività 4: eliminazione di generalizzazioni tra entità

- Una generalizzazione tra una entità padre **F** e le sottoentità $E_1, E_2, \ldots, E_n$, viene trattata come n relazioni E_1 **ISA F**,…, E_n **ISA F**, introducendo n relazioni binarie **ISA-E_1-F**,…, **ISA-E_n-F**.
- Per tenere conto delle proprietà delle generalizzazioni si aggiungono opportuni vincoli esterni, detti **vincoli di generalizzazione**:
  - il fatto che per ogni istanza **I** dello schema di partenza valga la proprietà $istanze(I,E_i) \cap istanze(I,E_k) = \emptyset$ per ogni $1 \leq i,k \leq n, i \neq k$ viene colto dal vincolo:

    *per ogni coppia (E_i,E_j) di entità figlie, ogni istanza di **F** partecipa **al più ad una** delle relazioni **ISA-E_i-F**, **ISA-E_j-F***
  - se la **generalizzazione** è **completa**, vale l'ulteriore proprietà $istanze(I,E_1) \cup \ldots \cup istanze(I,E_n) = istanze(I,F)$ nello schema di partenza e quindi il vincolo di generalizzazione diventa:

    *ogni istanza di **F** partecipa **esattamente ad una** delle relazioni **ISA-E_1-F,…, ISA-E_n-F***

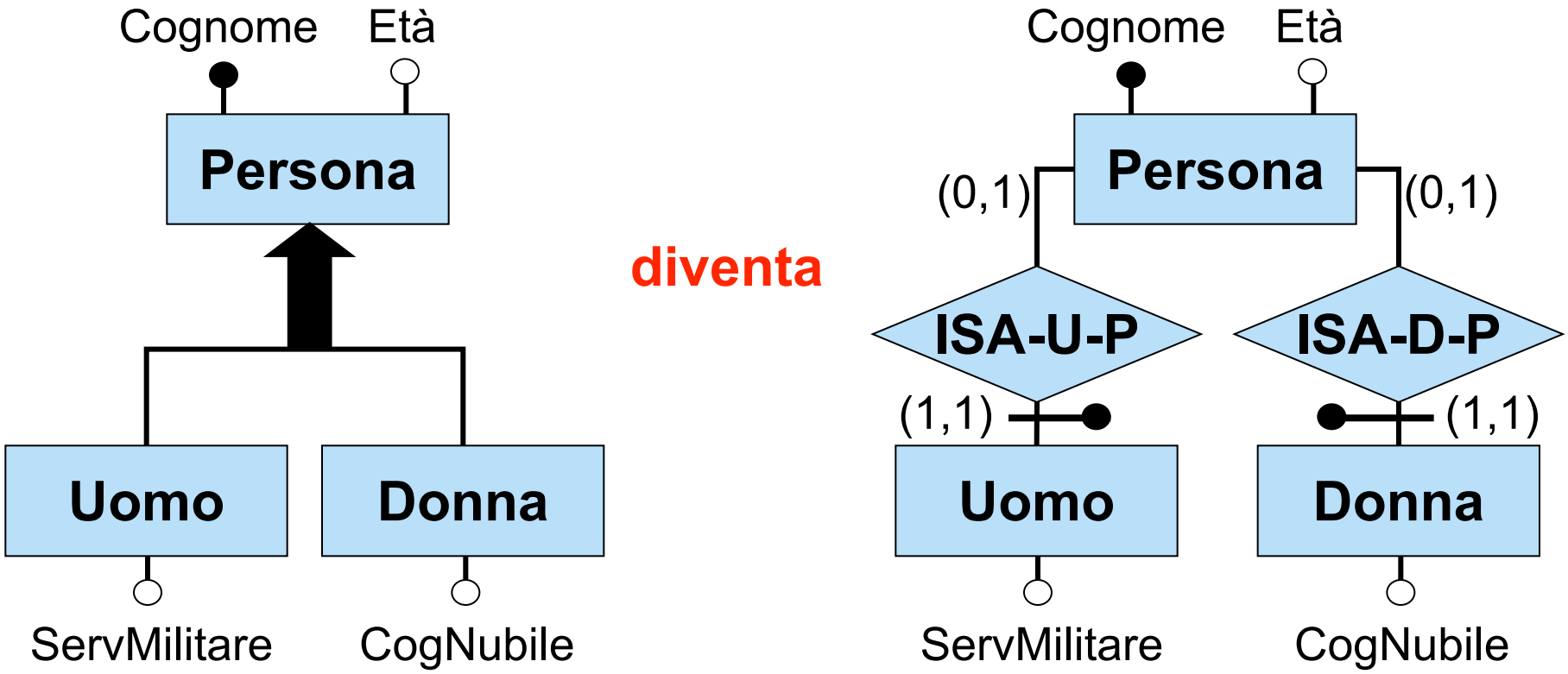
Eliminazione di generalizzazioni tra entità: esempio



Vincoli di generalizzazione non completa:

nessuna istanza di Persona partecipa sia a ISA-S-P sia a ISA-D-P

Eliminazione di generalizzazioni complete: esempio



Vincoli di generalizzazione completa:

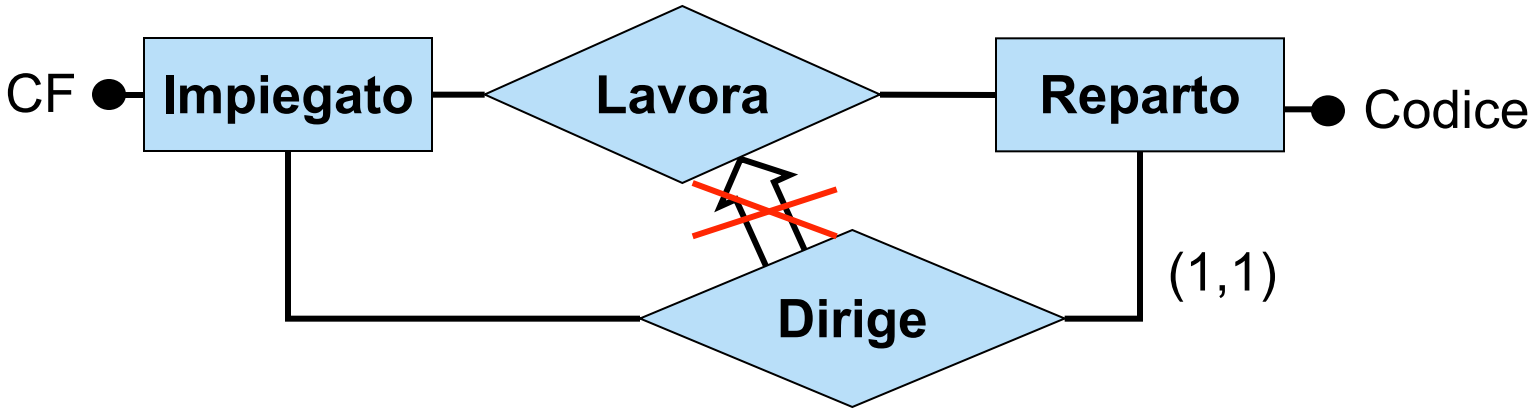
ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

Ristrutturazione – attività 4: eliminazione di ISA e generalizzazioni tra relazioni

Le **relazioni ISA e le generalizzazioni tra relazioni** vengono eliminate dallo schema e vengono espresse tramite opportuni vincoli esterni.

Nel caso in cui le relazioni in ISA (o nella generalizzazione) insistano su esattamente le **stesse entità per tutti i ruoli**, è immediato esprimere il vincolo esterno.

Esempio:

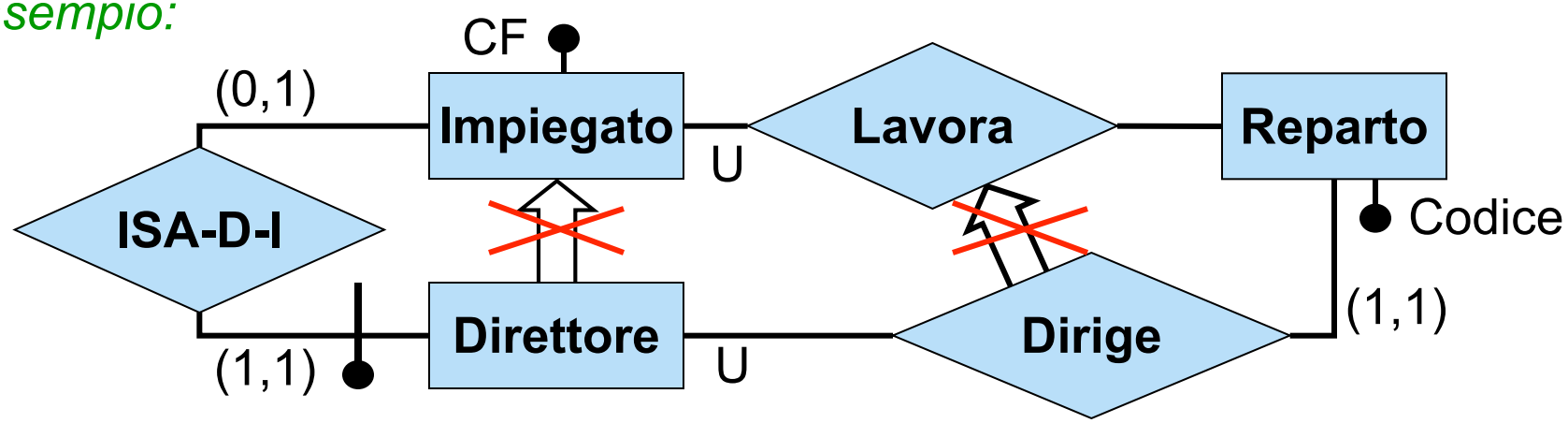


Vincolo esterno: ogni istanza di Dirige è anche un'istanza di Lavora.

Eliminazione di ISA e generalizzazioni tra relazioni

Nel caso in cui le relazioni in ISA (o nella generalizzazione) insistano su **entità diverse in qualche ruolo**, nell'esprimere il vincolo esterno bisogna tenere conto che nello schema ristrutturato entità diverse sono tra loro disgiunte.

Esempio:



Vincolo esterno: In ogni istanza I dello schema, per ogni istanza (U:*d*,Reparto:*r*) di Dirige in I, detta *i* l'istanza di Impiegato tale che (Direttore:*d*,Impiegato:*i*) è un'istanza di ISA-D-I in I (si noti che *i* esiste sempre ed è unica), si ha che (U:*i*,Reparto:*r*) è un'istanza di Lavora in I.

Ristrutturazione – attività 5: scelta degli identificatori principali

1. Scelta dell'identificatore principale di ogni entità
2. Scelta dell'identificatore principale di ogni relazione

Ogni identificatore principale di entità o relazione deve essere essenziale. Ricordiamo che un identificatore di W (entità o relazione) si dice essenziale se ogni insieme ottenuto da W togliendo un elemento non è più un identificatore per W.

Ristrutturazione – attività 5: scelta degli identificatori principali delle entità

Per **ogni entità** è necessario:

- individuare almeno un identificatore (se nello schema concettuale originario una entità non ne ha, occorre concordare con gli esperti del dominio un nuovo attributo – tipicamente un **codice** – che possa agire da identificatore)
- scegliere tra gli identificatori dell'entità un **identificatore principale**.

Criteri per la scelta dell'identificatore principale di entità:

- essenzialità (ogni identificatore principale deve essere essenziale)
- semplicità (cioè formato da pochi elementi)
- utilizzo nelle operazioni più frequenti o importanti
- preferenza per gli identificatori interni
- tra quelli identificatori esterni, privilegiare quelli sulle relazioni provenienti dalla eliminazione della relazione ISA
- se per un'entità nessuno degli identificatori soddisfa tali requisiti, è possibile introdurre un ulteriore attributo dell'entità (un **codice**, i cui valori sono speciali ed hanno l'unico scopo di identificare le istanze dell'entità).

In una entità con più identificatori, quello principale viene indicato nella documentazione associata allo schema ristrutturato: infatti, nel diagramma dello schema ristrutturato, in presenza di più identificatori per un'entità, denoteremo quello principale con un cerchio addizionale intorno al pallino dell'identificatore.

Cicli di identificazione esterna su entità

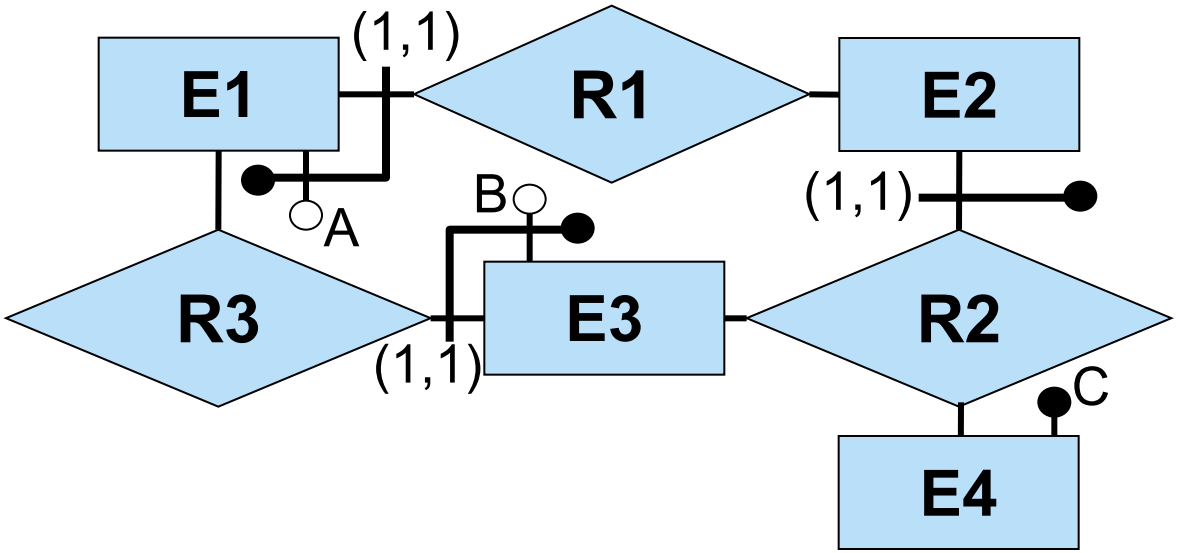
Nella scelta degli identificatori principali delle entità è necessario fare attenzione a non introdurre **cicli tra gli identificatori principali esterni**.

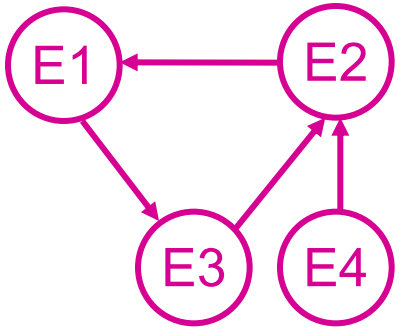
Definiamo il grafo degli identificatori (principali) esterni nel seguente modo:

- ad ogni entità del diagramma corrisponde un nodo
- c'è un arco dall'entità E all'entità F se e solo se E partecipa ad una relazione che è parte dell'identificatore (o che è identificatore) principale esterno di F.

Si ha un ciclo di identificazione esterna quando il grafo degli identificatori principali esterni contiene un ciclo.

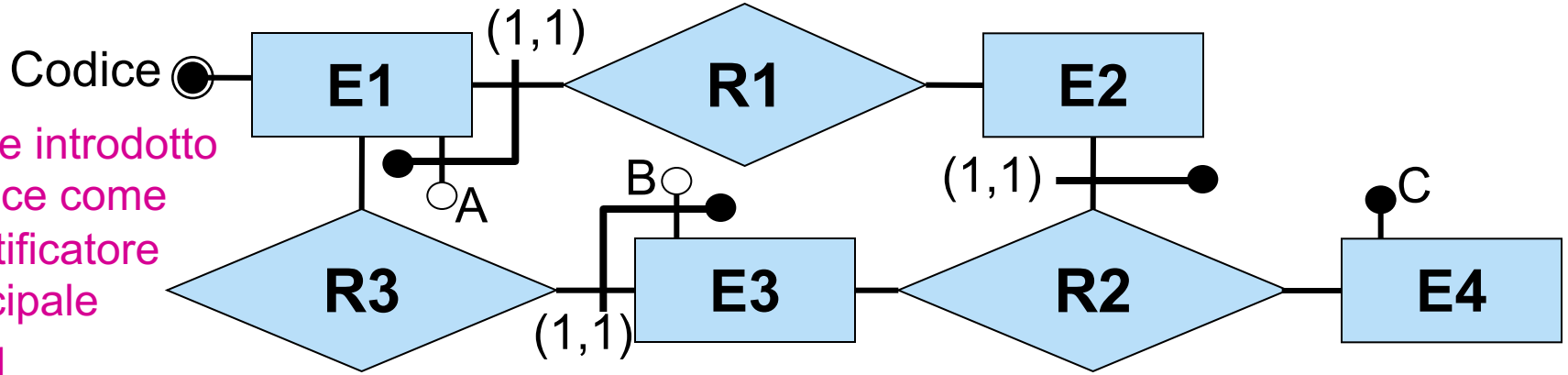
Cicli di identificazione esterna





grafo degli identificatori principali esterni

È necessario **spezzare i cicli di identificazione esterna** scegliendo per almeno una entità nel ciclo un identificatore principale diverso. Se non ci sono alternative, è necessario introdurre un opportuno codice.

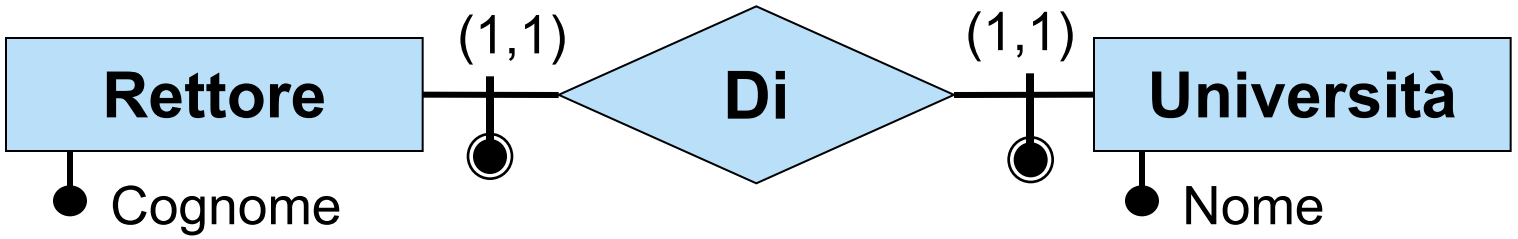


Viene introdotto Codice come identificatore principale di E1

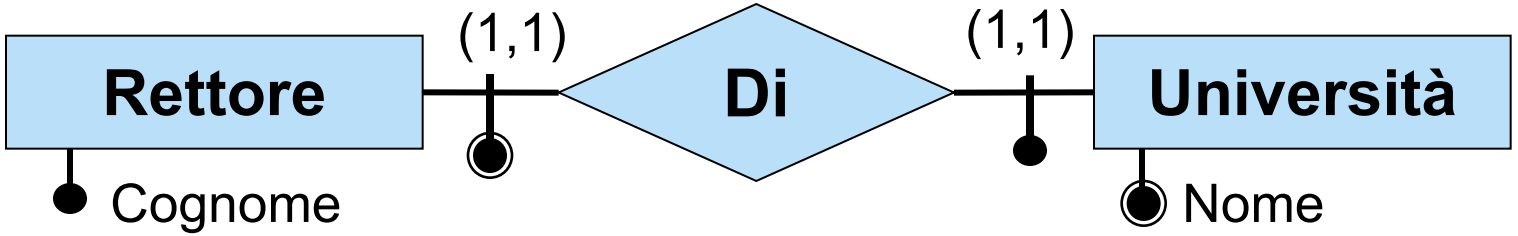
Cicli di identificazione esterna: esempio

Un caso significativo di ciclo di identificazione esterna è dato da due entità che si identificano a vicenda.

Esempio:



Abbiamo un ciclo di identificazione esterna, che deve essere spezzato. Una possibilità è la seguente:



Ristrutturazione – attività 5: scelta degli identificatori principali delle relazioni

Per **ogni relazione che non partecipa ad un identificatore principale di entità** è necessario scegliere, tra i suoi identificatori, l'**identificatore principale**.

Criteri per la scelta dell'identificatore principale di relazione:

- essenzialità (ogni identificatore principale deve essere essenziale)
- semplicità (cioè formato da pochi elementi),
- utilizzo nelle operazioni più frequenti o importanti,
- sulle relazioni provenienti dalla eliminazione della relazione ISA, privilegiare quello sul ruolo corrispondente alla entità figlia.

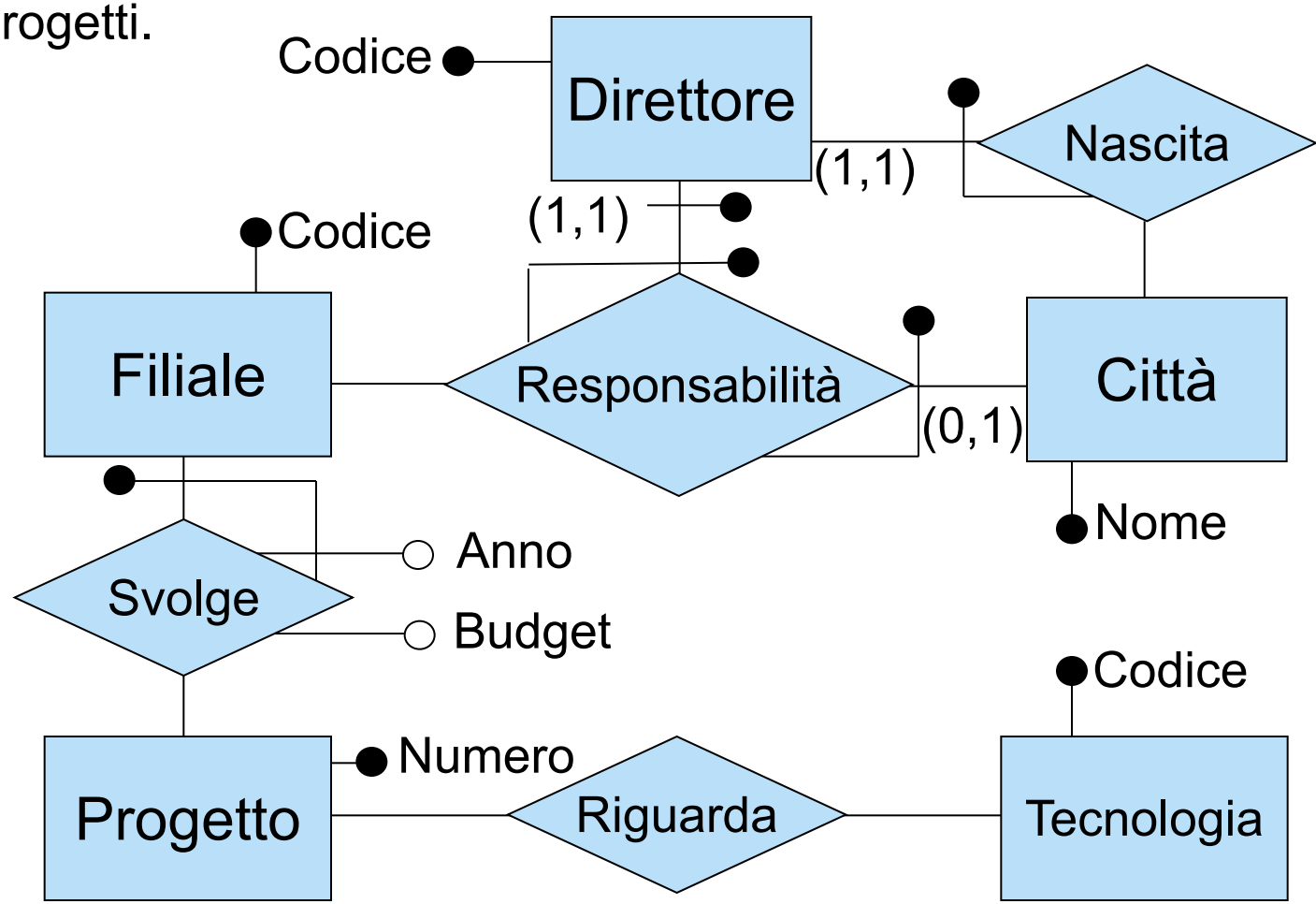
In una relazione con più identificatori, quello principale viene indicato nella documentazione associata allo schema ristrutturato: infatti, nel diagramma dello schema ristrutturato, in presenza di più identificatori per una relazione, denoteremo quello principale con un cerchio addizionale intorno al pallino dell'identificatore.

Scelta degli identificatori principali delle relazioni

1. Se una relazione R partecipa ad un identificatore principale esterno dell'entità E, allora l'identificatore di E di fatto rappresenta già l'identificatore principale di R. In questo caso la scelta è già implicita e quindi non dobbiamo fare alcuna scelta dell'identificatore principale di R
2. Se una relazione R non partecipa ad alcun identificatore esterno di entità e non ha alcun identificatore indicato esplicitamente diverso da quello implicito (ossia quello formato da tutti i ruoli di R), allora l'identificatore implicito diventa l'identificatore principale della relazione R (ed in questo caso possiamo scegliere o di indicarlo o di non indicarlo nel diagramma dello schema ristrutturato)
3. Se una relazione non partecipa ad alcun identificatore principale esterno di entità ed ha uno o più identificatori indicati esplicitamente che siano diversi da quello implicito, allora se ne deve scegliere uno come identificatore principale (al limite scegliendo quello implicito) ed indicarlo nel modo convenuto. Se scegliamo l'identificatore implicito come principale, questo sarà il caso in cui andrà indicato nel diagramma dello schema ristrutturato

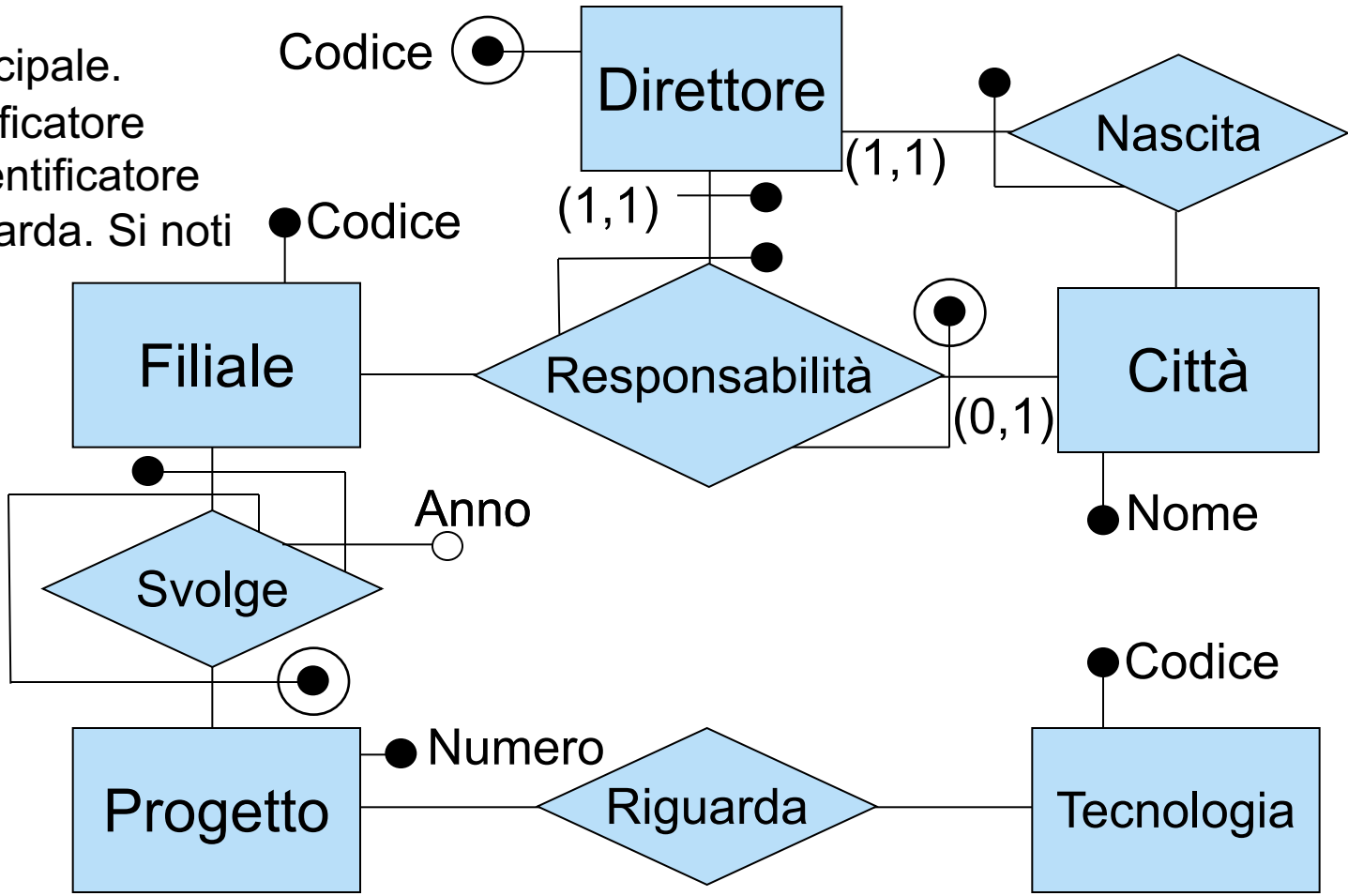
Esempio: identificatori principali delle relazioni

Consideriamo il seguente schema ristrutturato e supponiamo che le operazioni usino principalmente il codice per accedere ai direttori, le città per accedere a Responsabilità e la combinazione di filiale e numero di progetto (piuttosto che l'anno) per accedere alle informazioni sullo svolgimento dei progetti.



Esempio: identificatori principali delle relazioni

Scegliamo gli identificatori principali Codice per Direttore e Città per Responsabilità. Scegliamo l'identificatore implicito come identificatore principale di Svolge; esso viene indicato nel diagramma dello schema ristrutturato perché altrimenti lasceremmo indicato il solo identificatore (Anno, Filiale) che quindi sarebbe interpretato come Identificatore principale. Scegliamo l'identificatore implicito come identificatore principale di Riguarda. Si noti che nel caso della relazione Riguarda non serve indicare l'identificatore nel diagramma dello schema ristrutturato, perché non ci sono identificatori espliciti per tale relazione.



Ristrutturazione – attività 6: specifica degli ulteriori vincoli esterni

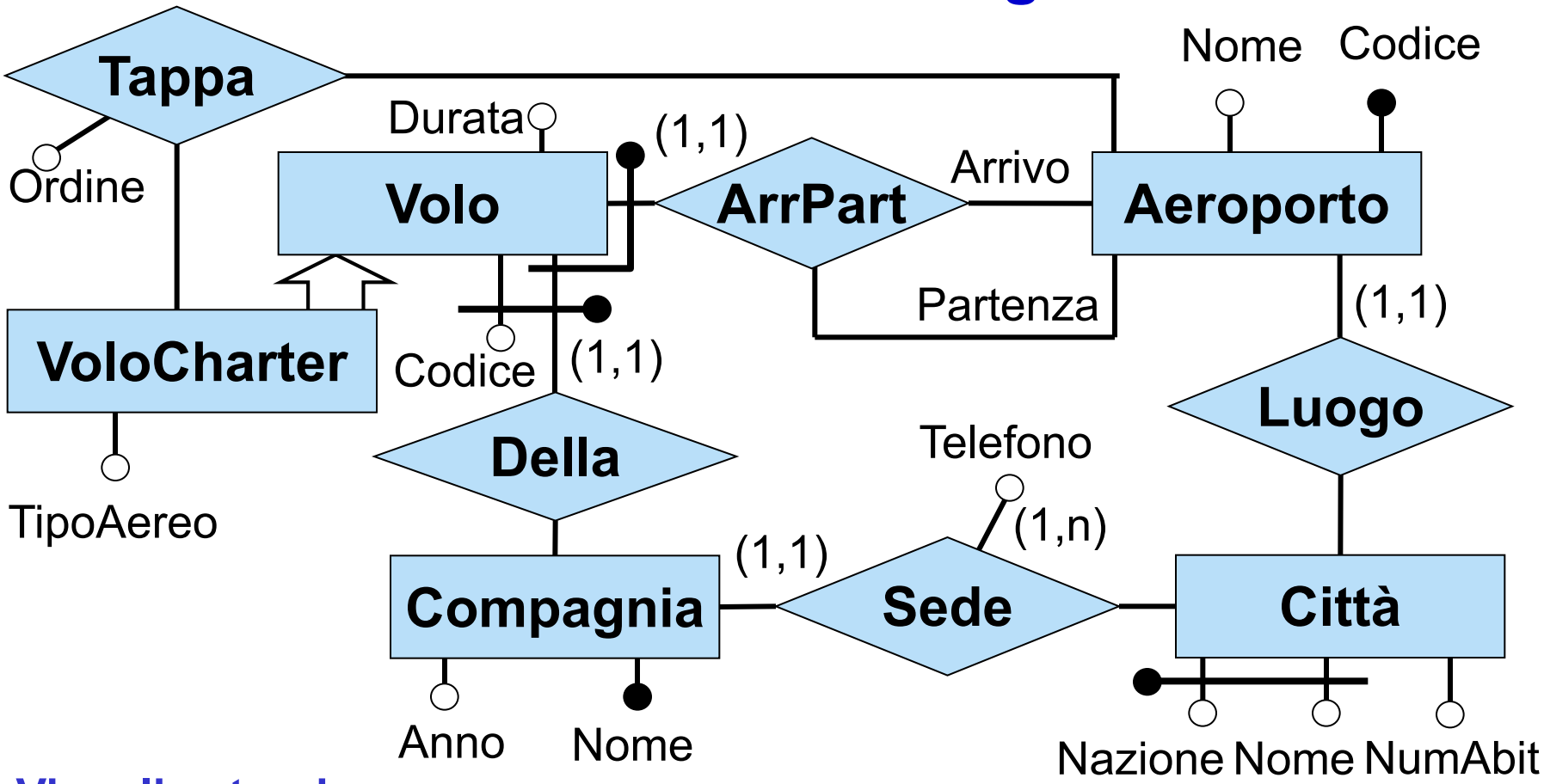
- È necessario riformulare tutti i vincoli esterni dello schema originario in termini dello schema ristrutturato.
  - mettere in evidenza anche i vincoli impliciti (ovvero che sono conseguenza di altri vincoli), se non è ancora stato fatto

    Es. vincoli di identificazione esterna per entità che partecipano a relazioni (1,1) con (0,1) o (1,1) in altro ruolo
- Si devono aggiungere eventuali vincoli derivanti dalla ristrutturazione:
  - vincoli derivanti da attributi composti opzionali
  - vincoli per due entità che erano in ISA con una stessa entità padre e che hanno attributi in comune
  - vincoli di generalizzazione (disgiuntezza, completezza)
  - vincoli dovuti agli identificatori non principali (se non sono più rappresentati nello schema)

Riassunto sulla ristrutturazione

0. Preparazione iniziale dello schema ER
1. Analisi delle ridondanze (si tiene conto dell'efficienza)
2. Eliminazione degli attributi multivalore
3. Eliminazione degli attributi composti (eventuale vincolo (0,1) diventa un vincolo esterno sugli attributi componenti)
4. Eliminazione delle ISA e delle generalizzazioni
   - vincoli per stesso attributo di entità figlie della stessa entità padre
   - vincoli di generalizzazione (disgiuntezza e completezza)
   - si noti che tutte le entità diventano disgiunte
5. Scelta degli identificatori principali di entità e relazioni
   - tutte le entità devono avere un identificatore (eventualmente, introdurre opprtuni attributi di tipo codice)
6. Specifica degli ulteriori vincoli esterni
   - vincoli derivanti dalla ristrutturazione
   - riformulazione dei vincoli esterni dello schema originario

Esercizio 3: ristrutturare il seguente schema

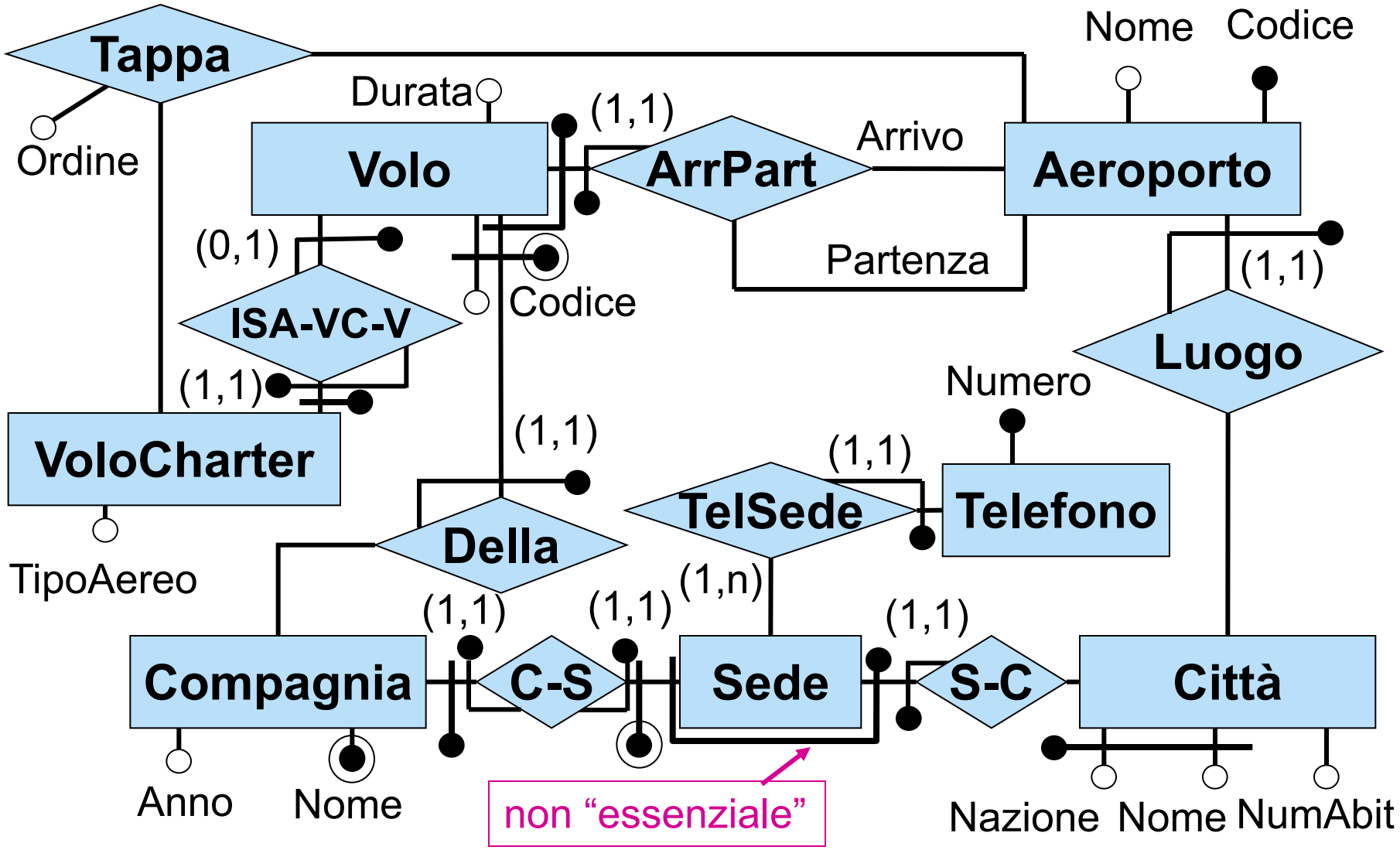


Vincoli esterni:

1) per ogni v in VoloCharter, se $(v,a_1),...,(v,a_n)$ sono tutte le coppie in Tappa alle quali partecipa v, e se $o_1,...,o_n$ sono i valori assegnati a tali coppie dall'attributo Ordine, allora per i=1,...,n esiste un o_j tale che o_j=i.

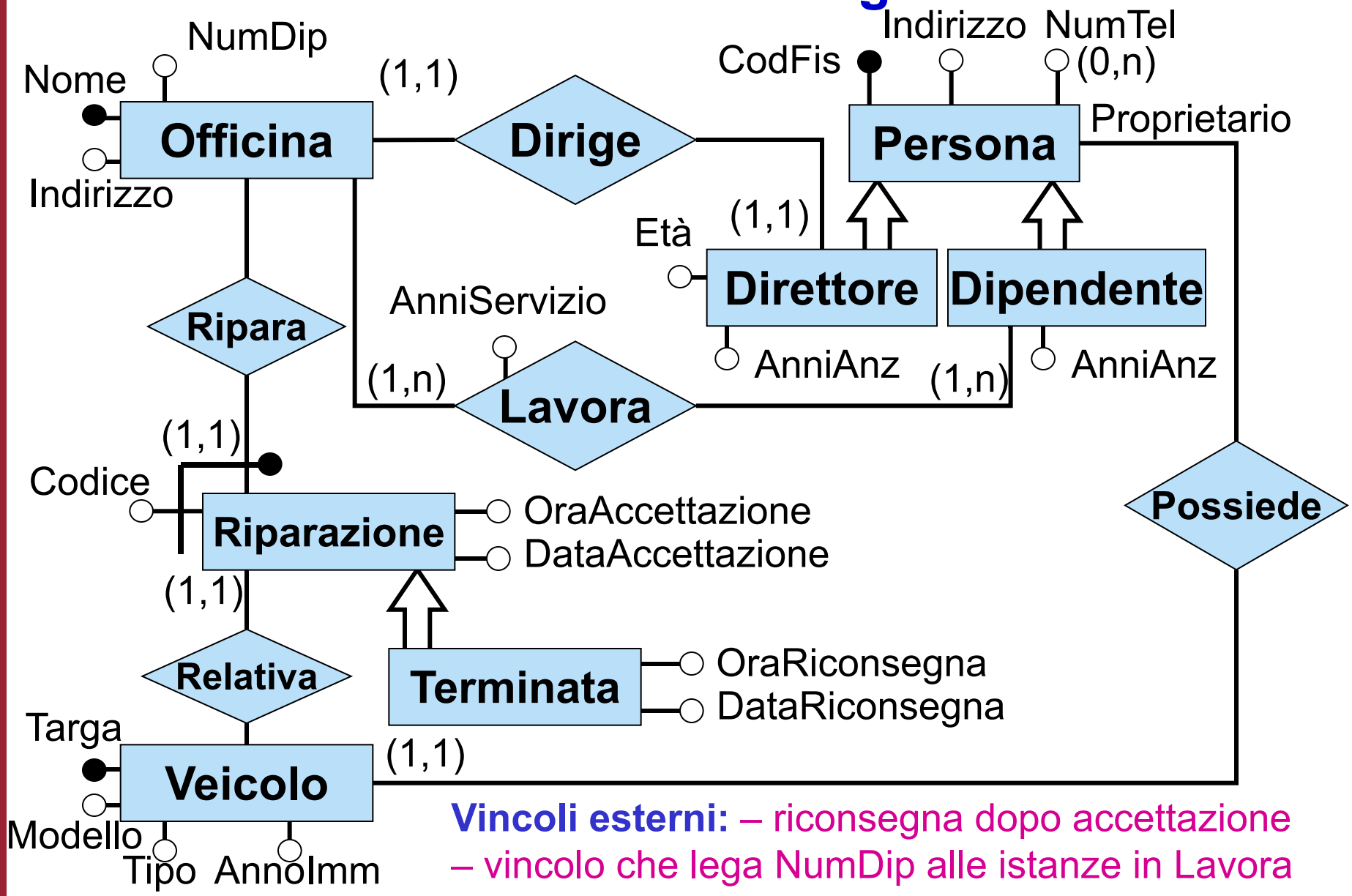
2) un numero di telefono è di una sola sede.

Esercizio 3: soluzione



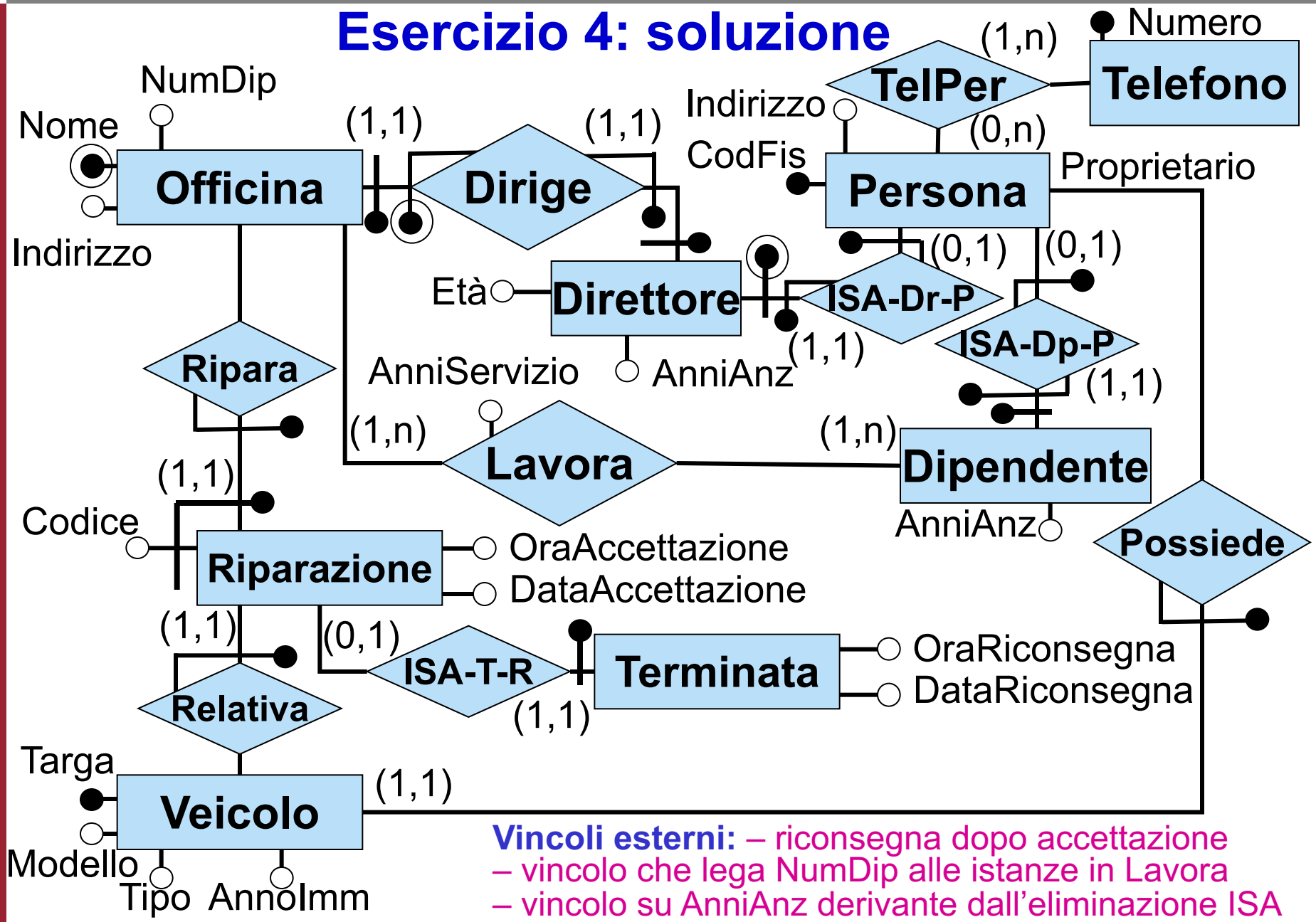
Vincolo esterno: 1) vincolo su Ordine in Tappa (2 è diventato interno allo schema)

Esercizio 4: ristrutturare il seguente schema



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora

Esercizio 4: soluzione



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora
– vincolo su AnniAnz derivante dall'eliminazione ISA

6. La progettazione logica

6.3 traduzione diretta nel modello relazionale

1. introduzione alla progettazione logica
2. ristrutturazione dello schema ER
3. **traduzione diretta nel modello relazionale**
4. ristrutturazione dello schema logico

Proprietà dello schema ristrutturato

La fase di ristrutturazione ha prodotto uno schema ER ristrutturato con le seguenti proprietà:

- preserva la semantica dello schema originale. Intuitivamente, esiste una funzione che associa ad ogni istanza dello schema originale un'opportuna istanza dello schema ristrutturato e viceversa
- può contenere delle ridondanze, ma sono volute per motivi di efficienza e sono comunque documentate
- non contiene attributi multivalore
- non contiene attributi composti
- non contiene ISA o generalizzazione (né tra entità, né tra relazioni); quindi tutte le entità sono disgiunte a coppie
- tutte le entità e tutte le relazioni hanno un unico identificatore principale

Lo schema ristrutturato è il punto di partenza per la traduzione nel modello relazionale.

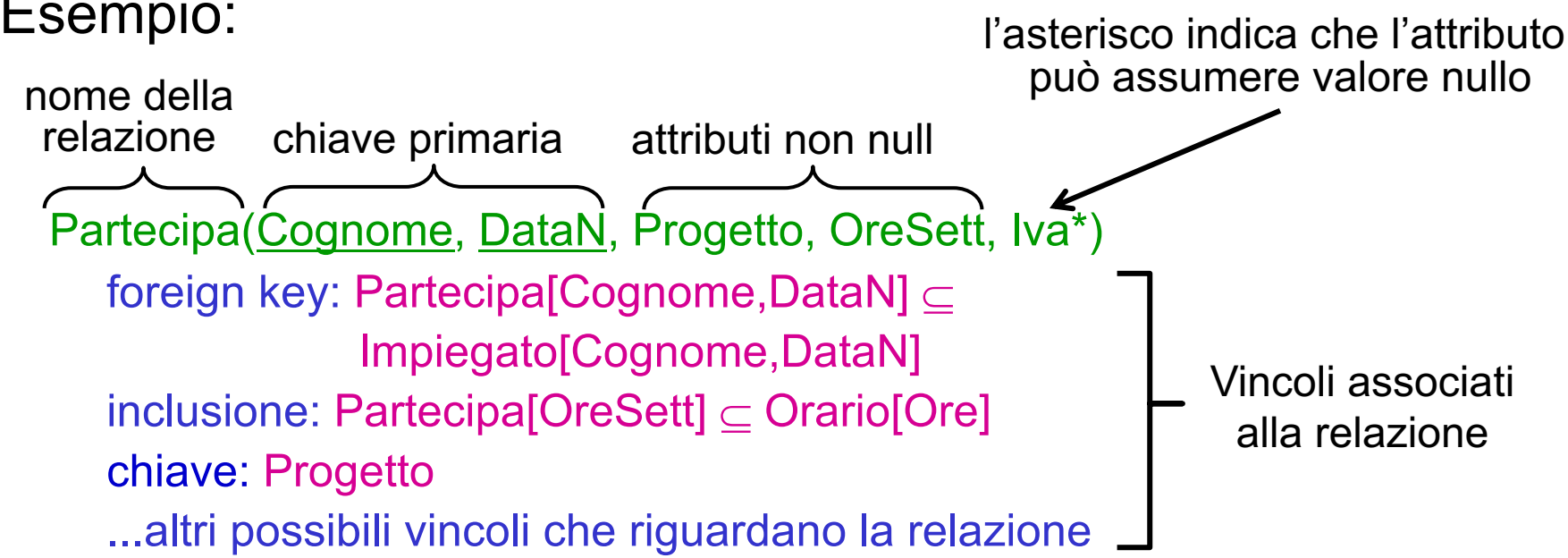
Traduzione diretta

- La traduzione diretta ha lo scopo di tradurre lo schema ER ristrutturato (con vincoli) in uno schema relazionale con vincoli che rappresenti le stesse informazioni
- Viene eseguita seguendo un metodo automatico, cioè che di fatto non richiede di effettuare scelte, in quanto si basa sulle scelte fatte in fase di ristrutturazione
- Produce uno schema logico di massima, che può essere accettabile, ma che può richiedere successive ristrutturazioni, in particolare nella fase di ristrutturazione dello schema logico
- Consiste delle seguenti **attività**:
  (1) traduzione delle **entità** in relazioni dello schema logico, con relativi vincoli
  (2) traduzione delle **relazioni** dello schema ER in relazioni dello schema logico, con relativi vincoli
  (3) traduzione dei **vincoli esterni** in vincoli dello schema logico
  (4) riformulazione di **operazioni** e **specifiche** sul carico applicativo in termini dello schema logico

Come al solito, per distinguere tra le due accezioni di relazione, useremo il termine ER-relazione per denotare le relazioni dello schema ER

Notazione

- Nella fase di traduzione diretta ed in quella di ristrutturazione dello schema logico, esprimeremo gli schemi relazionali mediante una notazione che prevede di descrivere le relazioni con nome e attributi, ed i vincoli ad esse associati in forma testuale. Illustriamo la notazione con un esempio.
- Esempio:



- Ovviamente, questa notazione si può tradurre in termini di "create table" in SQL. Si lascia allo studente la verifica di come effettuare la traduzione in SQL

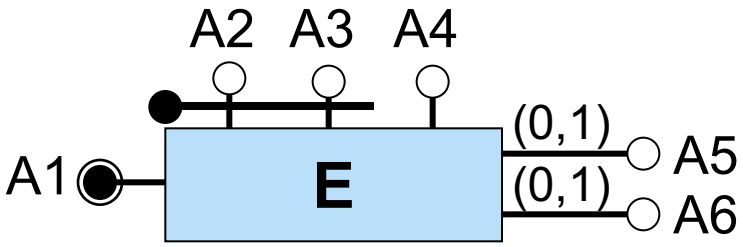
(1) Traduzione di entità: regole generali

- Ogni entità **E** dello schema ER viene tradotta in una relazione R_E dello schema relazionale
- Gli **attributi** della relazione R_E sono:
  - gli attributi dell'entità **E** (tutti not null, tranne quelli con cardinalità (0,1))
  - gli attributi derivanti dall'accorpamento di ER-relazioni in R_E – per ogni ER-relazione **Q** accorpata in R_E (si veda qui sotto) vengono aggiunti ad R_E:
    - le chiavi primarie delle relazioni che corrispondono alle altre entità che partecipano a **Q**
    - gli eventuali attributi della ER-relazione **Q**
- Una ER-relazione **Q** viene **accorpata** in R_E quando uno o più ruoli di **Q** partecipano all'identificatore principale (esterno) di **E** (si noti che in questo caso **E** partecipa a **Q** con cardinalità (1,1), ed inoltre **E** è l'unica entità per cui un ruolo di **Q** partecipa all'identificatore principale esterno, altrimenti ci sarebbe un ciclo di identificazione principale esterna)
- La **chiave primaria** di R_E è determinata in base all'identificatore principale di **E** (se interno, formato dagli attributi di R_E, altrimenti formato da proprietà derivanti dall'identificazione esterna)
- Gli altri vincoli di identificazione di **E** verranno tipicamente tradotti in **vincoli di chiave** (non primaria) su R_E
- A seconda dei casi, possono essere necessari ulteriori vincoli (in particolare, vincoli di foregn key per rappresentare la partecipazione obbligatoria della entità alle relazioni)

(1) Traduzione di entità senza accorpamento

Consideriamo per ora **un'entità per cui non si effettua accorpamento di relazioni** (in particolare, un'entità che non ha identificatori esterni).

- L'entità si traduce in una **relazione** dello schema relazionale
- Gli **attributi** della relazione corrispondente all'entità sono quelli dell'entità;
  - se un attributo è opzionale diventa un attributo della relazione che può assumere valore nullo (nella nostra notazione per lo schema logico, tali attributi sono indicati con *)
  - altrimenti l'attributo non può assumere valore nullo
- L'identificatore principale dell'entità si traduce nella **chiave primaria** della relazione
- Gli altri identificatori interni si traducono in **chiavi** della relazione
- Ricordarsi dei vincoli esterni per identificatori opzionali correlati (derivanti da attributi composti opzionali)



Vincoli esterni:

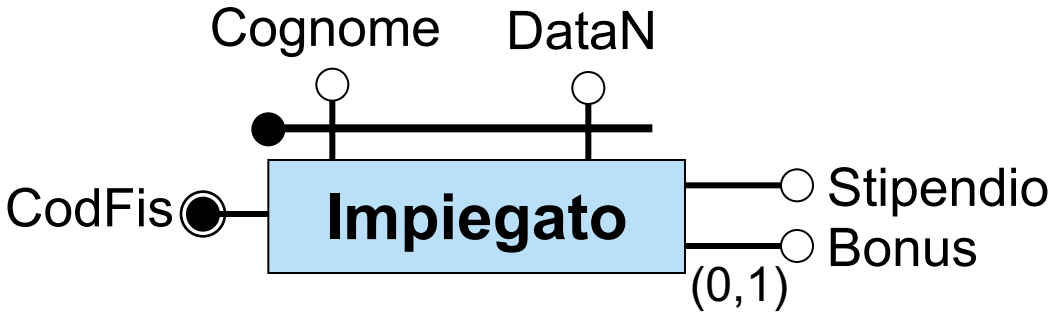
- *per ogni istanza di E, A5 è definito se e solo se lo è anche A6*

E(A1, A2, A3, A4, A5*, A6*)

chiave: A2, A3

vincolo di tupla: A5 è NULL se e solo se A6 è NULL

(1) Traduzione di entità senza accorpamento: esempio



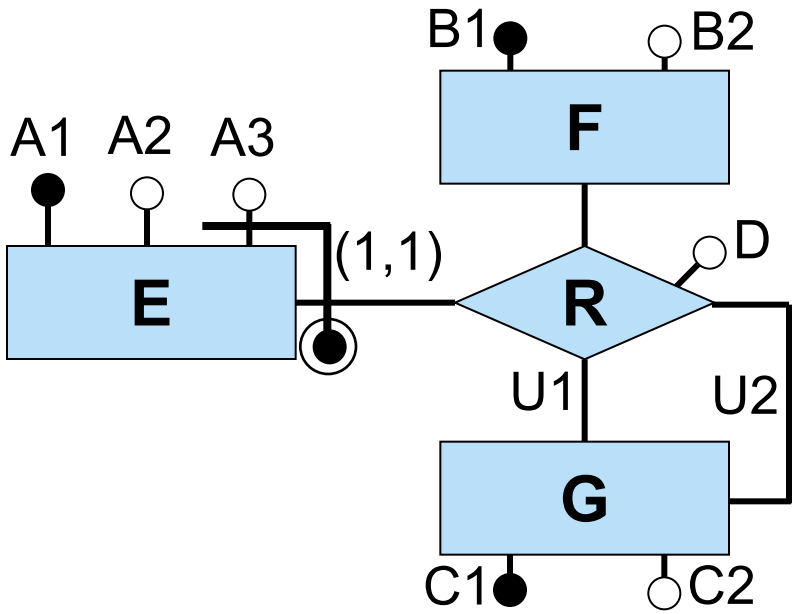
Impiegato(<u>CodFis</u>, Cognome, DataN, Stipendio, Bonus*)

chiave: Cognome, DataN

(1) Traduzione di entità con accorpamento: caso 1

Consideriamo il caso in cui una ER-relazione **R** è **parte dell'identificatore principale esterno di un'entità E con ruolo U**, e in tutti gli altri ruoli di **R** la cardinalità è (0,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E**. Questo significa che tutti gli attributi della ER-relazione e le chiavi primarie delle altre entità partecipanti diventano attributi della relazione che corrisponde all'entità **E**. Tali chiavi primarie contribuiscono (insieme agli eventuali altri attributi che formano l'identificatore principale di E) alla chiave primaria della relazione RE.
- Si noti che eventuali altri identificatori dell'entità in cui la relazione è stata accorpata si traducono in vincoli di chiave sull'entità.



E(A1, A2, A3, F, U1, U2, D)

foreign key: E[F] $\subseteq$ F[B1]

foreign key: E[U1] $\subseteq$ G[C1]

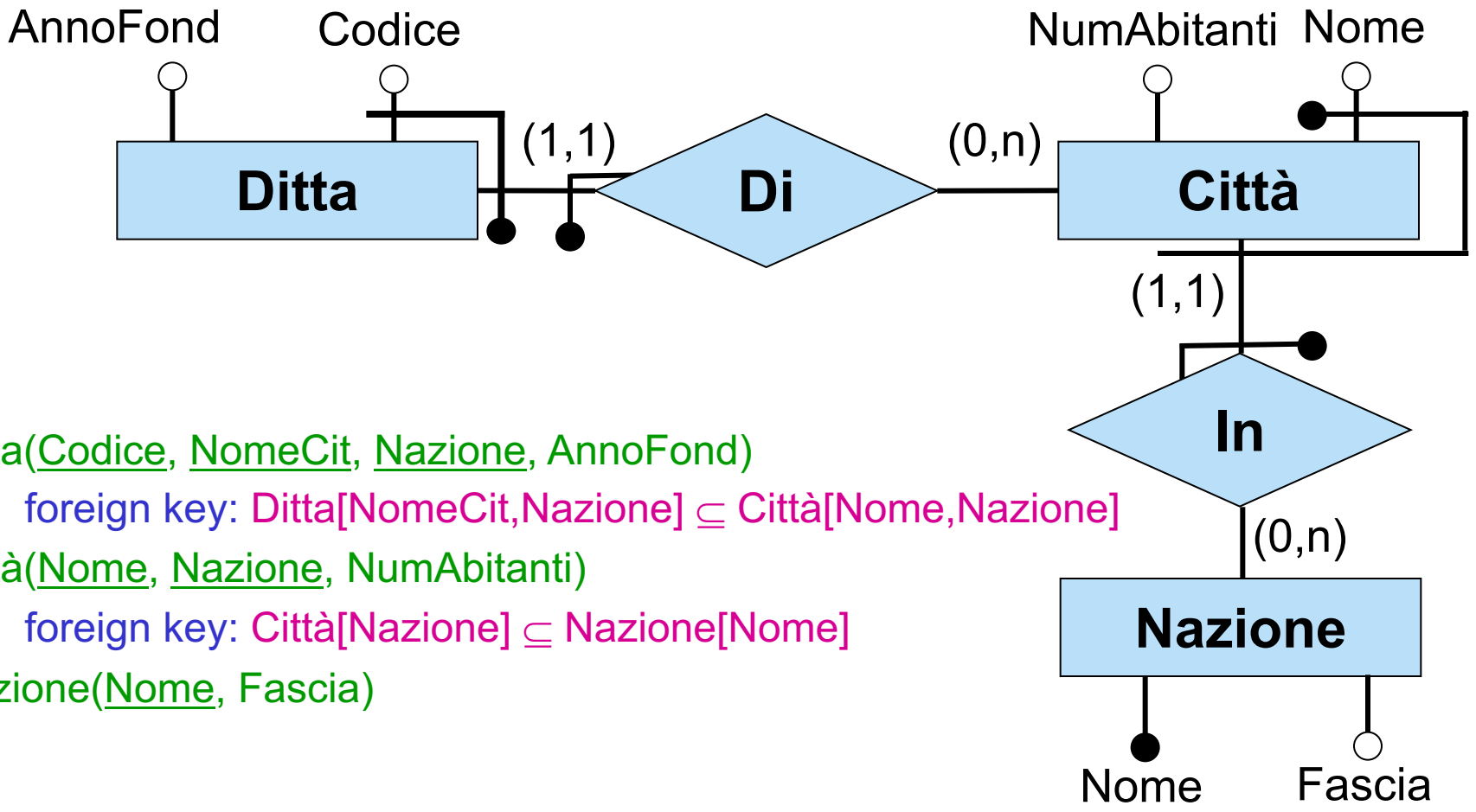
foreign key: E[U2] $\subseteq$ G[C1]

chiave: A1

F(B1, B2)

G(C1, C2)

(1) Traduzione di entità con accorpamento - caso 1: esempio



Ditta(Codice, NomeCit, Nazione, AnnoFond)

foreign key: Ditta[NomeCit,Nazione] $\subseteq$ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

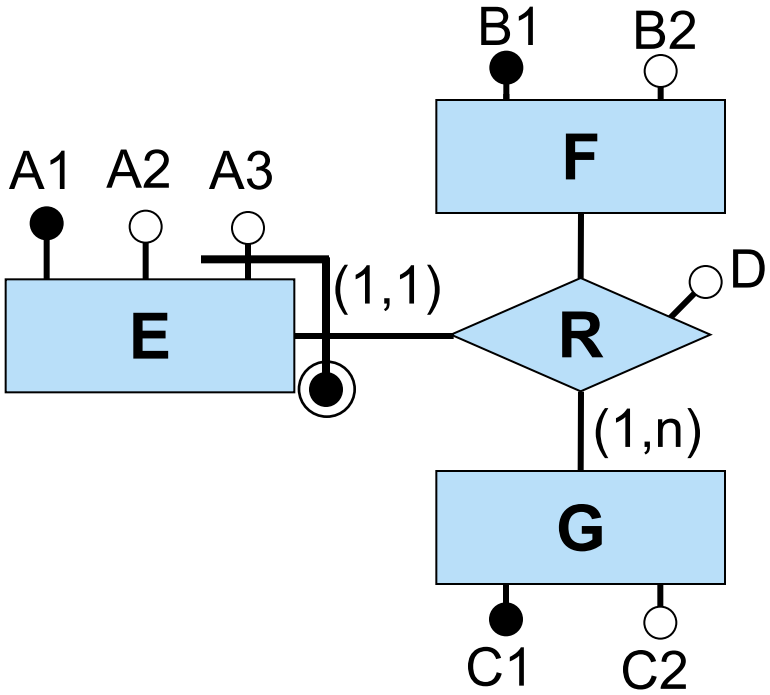
foreign key: Città[Nazione] $\subseteq$ Nazione[Nome]

Nazione(Nome, Fascia)

(1) Traduzione di entità con accorpamento: caso 2

Consideriamo il caso in cui una ER-relazione **R** è **parte dell'identificatore principale esterno di un'entità E con ruolo U**, ed in alcuni degli altri ruoli di **R** la cardinalità è (1,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E** in modo analogo al caso 1
- Ma in questo caso, per ogni ruolo **U** con cardinalità (1,n), si aggiunge un vincolo di inclusione tra la chiave primaria dell'entità corrispondente al ruolo **U** e l'attributo (o gli attributi, se la chiave è composta) di **E** corrispondente ad **U**



E(A1, A2, A3, F, G, D)

foreign key: E[F] $\subseteq$ F[B1]

foreign key: E[G] $\subseteq$ G[C1]

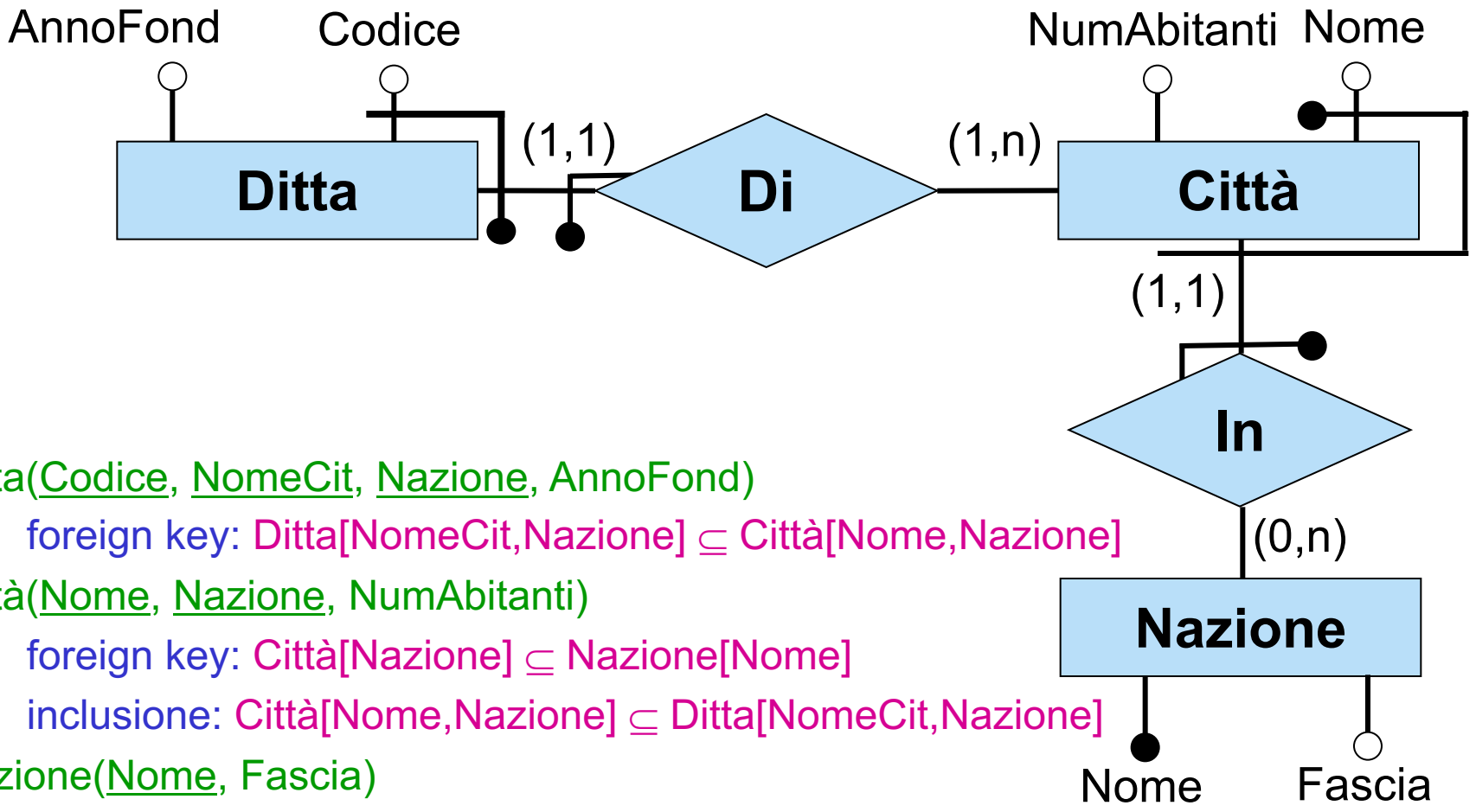
chiave: A1

F(B1, B2)

G(C1, C2)

inclusione: G[C1] $\subseteq$ E[G]

(1) Traduzione di entità con accorpamento - caso 2: esempio



Ditta(Codice, NomeCit, Nazione, AnnoFond)

foreign key: Ditta[NomeCit,Nazione] $\subseteq$ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

foreign key: Città[Nazione] $\subseteq$ Nazione[Nome]

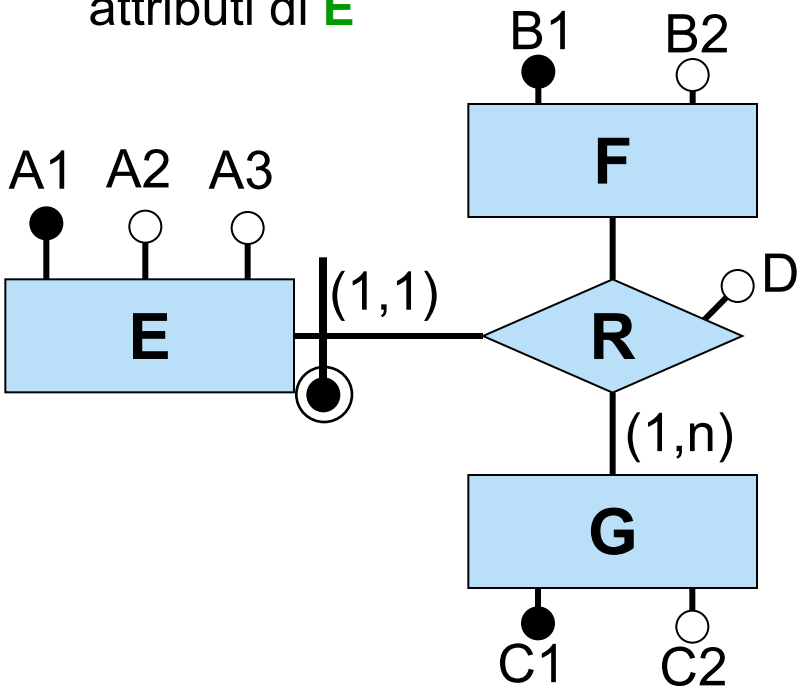
inclusione: Città[Nome,Nazione] $\subseteq$ Ditta[NomeCit,Nazione]

Nazione(Nome, Fascia)

(1) Traduzione di entità con accorpamento: caso 3

Consideriamo il caso in cui una ER-relazione **R** è **identificatore principale esterno** (non "parte" di identificatore) **di un'entità E con ruolo U**, ed in alcuni degli altri ruoli di **R** la cardinalità è (1,n).

- La ER-relazione **R** viene **accorpata** nell'entità **E** ed opportuni vincoli di inclusione riflettono la cardinalità minima pari ad 1, in modo analogo ai casi 1 e 2
- La differenza è che in questo caso la chiave primaria di **E** non comprende attributi di **E**



E(A1, A2, A3, F, G, D)

foreign key: E[F] $\subseteq$ F[B1]

foreign key: E[G] $\subseteq$ G[C1]

chiave: A1

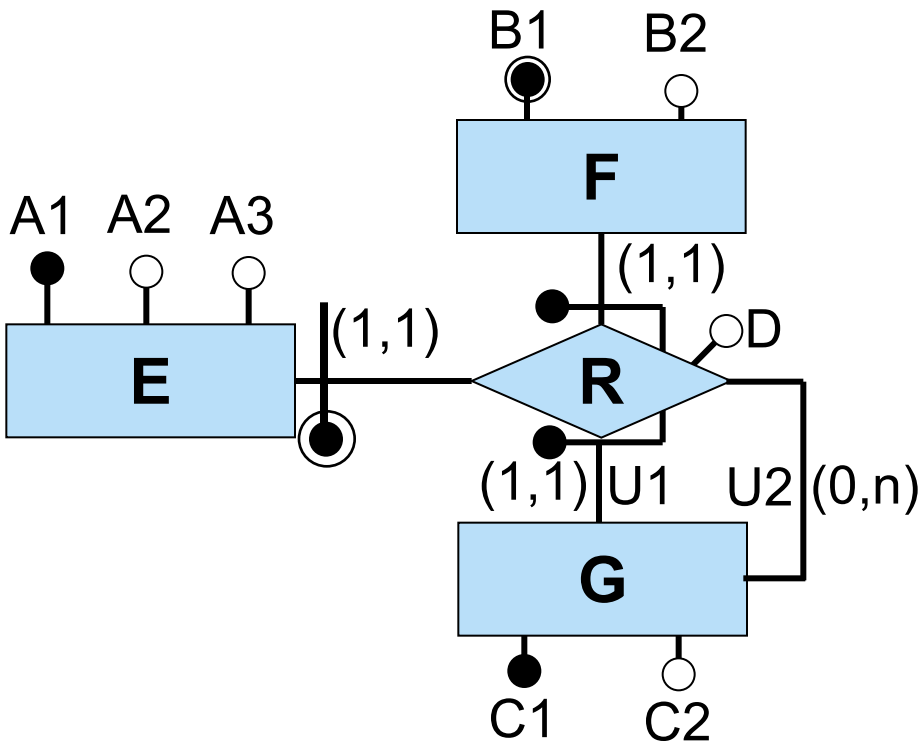
F(B1, B2)

G(C1, C2)

inclusione: G[C1] $\subseteq$ E[G]

(1) Traduzione di entità con accorpamento: caso 4

Consideriamo il caso in cui una ER-relazione **R** è **identificatore principale esterno di un'entità E con ruolo U**, e in almeno uno degli altri ruoli di **R** la cardinalità massima è 1. Si noti che in questo caso **R** è l'identificatore principale (non "parte" di identificatore) di **E**. La ER-relazione **R** viene **accorpata** nell'entità **E** in modo analogo ai casi precedenti, ma in questo caso la chiave primaria della relazione che rappresenta **E** deve essere scelta tra le chiavi primarie delle altre entità coinvolte in **R** i cui ruoli hanno cardinalità massima 1.



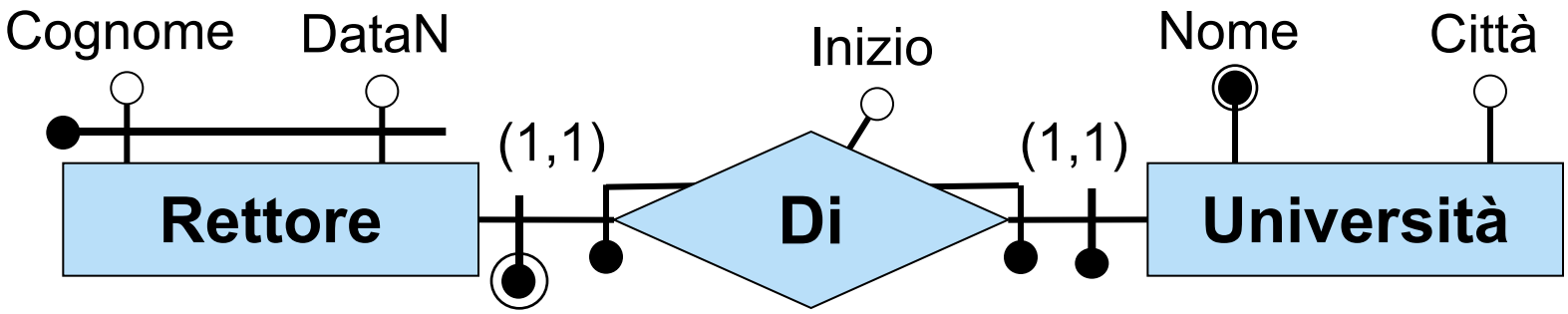
Supponiamo di scegliere la chiave primaria di F come chiave primaria di E (U1 diventa quindi chiave non primaria di E):

E(A1, A2, A3, F, U1, U2, D)
- foreign key: E[F] ⊆ F[B1]
- foreign key: E[U1] ⊆ G[C1]
- foreign key: E[U2] ⊆ G[C1]
- chiave: A1
- chiave: U1

F(B1, B2)
- foreign key: F[B1] ⊆ E[F]

G(C1, C2)
- foreign key: G[C1] ⊆ E[U1]

(1) Traduzione di entità con accorpamento – caso 4: esempio



Rettore(Cognome, DataN, Inizio, Università)

foreign key: Rettore[Università] $\subseteq$ Università[Nome]

chiave: Cognome, DataN

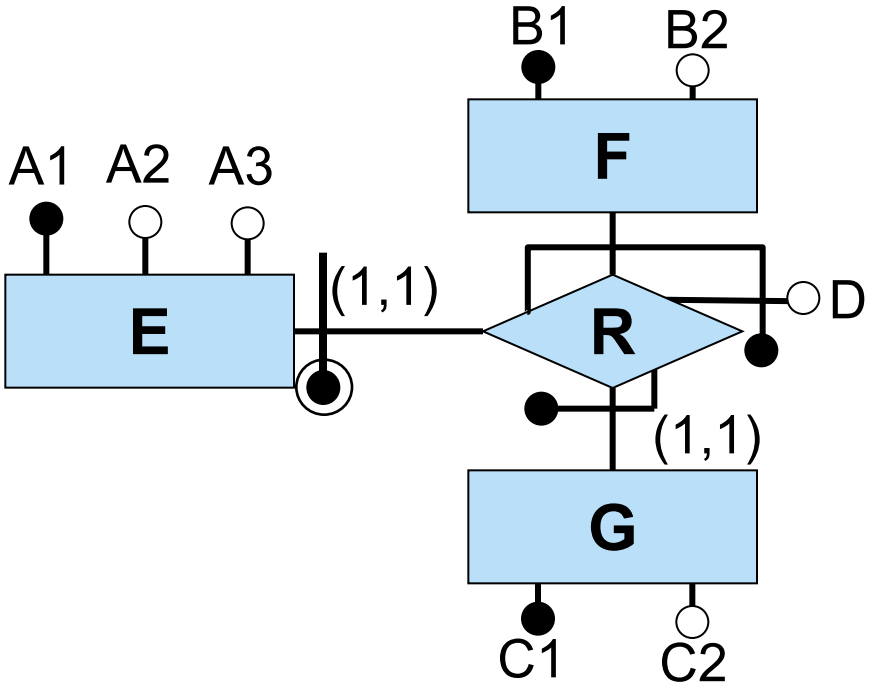
Università(Nome, Città)

foreign key: Università[Nome] $\subseteq$ Rettore[Università]

(1) Traduzione di entità con accorpamento: caso 5

Consideriamo il caso in cui una ER-relazione **R** è **identificatore principale esterno** (non "parte" di identificatore) **di un'entità E con ruolo U**, in alcuni degli altri ruoli di **R** la cardinalità è (1,1), ed **R** ha uno o più identificatori, magari costituiti anche da attributi.

- La ER-relazione **R** viene **accorpata** nell'entità **E**, in modo analogo ai casi 1, 2 e 3. Scegliamo **G** come chiave primaria di **E**,
- Anche in questo caso la tabella **E** ha chiavi in più (nel caso qui sotto la chiave F,D), che provengono dagli identificatori di **R**



E(A1, A2, A3, F, G, D)

foreign key: E[F] $\subseteq$ F[B1]

foreign key: E[G] $\subseteq$ G[C1]

chiave: A1

chiave: F,D

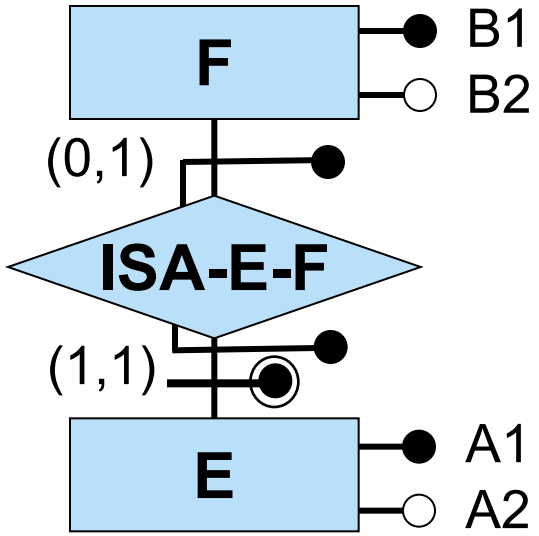
F(B1, B2)

G(C1, C2)

foreign key: G[C1] $\subseteq$ E[G]

(1) Accorpamento di ER-relazione derivante da ISA o generalizzazione

Un caso importante di ER-relazione che è identificatore principale esterno di un'entità è quello derivante dalla ristrutturazione di una relazione ISA o di una generalizzazione nello schema ER originale.



E(A1, A2, B1)

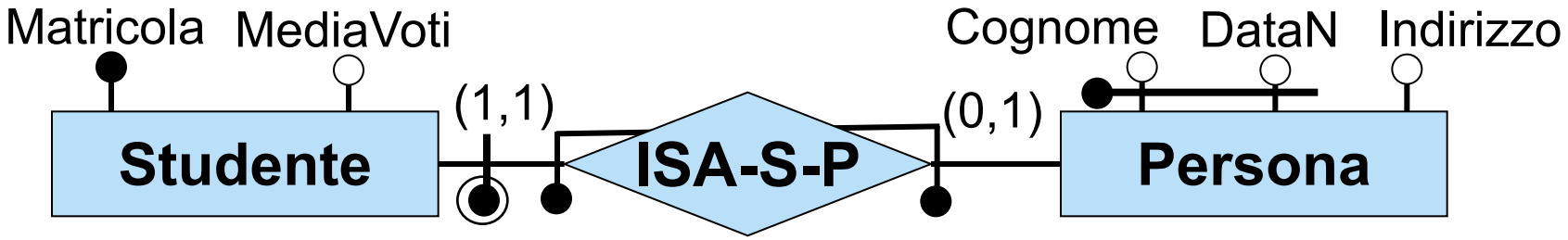
foreign key: E[B1] $\subseteq$ F[B1]

chiave: A1

F(B1, B2)

Si noti come la traduzione della parte di schema ER che si ottiene dalla ristrutturazione di **E ISA F** corrisponda ad aggiungere agli attributi della tabella **E** la chiave primaria della tabella **F**, ed a rendere tali attributi anche chiave primaria della tabella **E**. Il vincolo derivante dall'ISA dello schema ER originario diventa quindi un vincolo di foreign key dello schema logico.

(1) Accorpamento di ER-relazione derivante da ISA: esempio



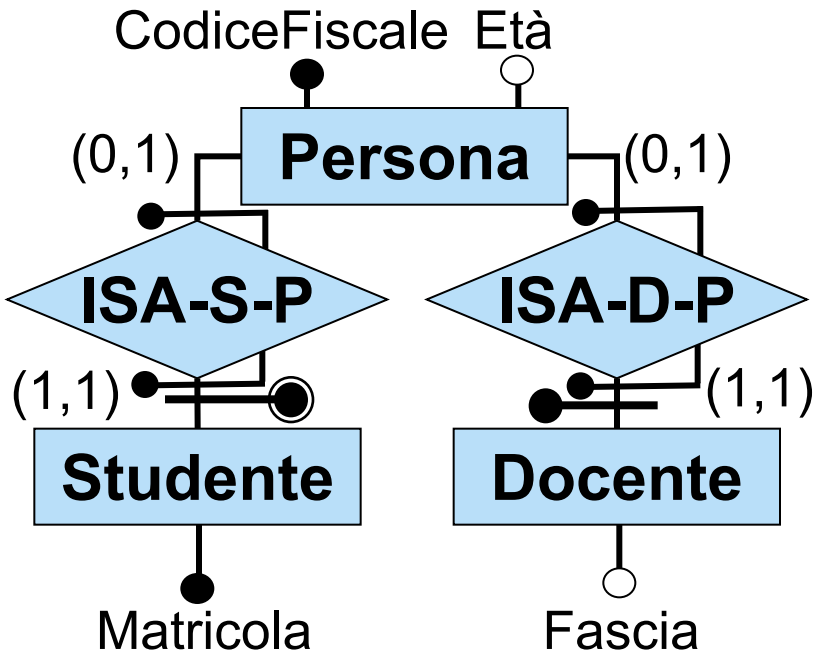
Studente(Cognome, DataN, Matricola, MediaVoti)

foreign key: Studente[Cognome,DataN] $\subseteq$ Persona[Cognome,DataN]

chiave: Matricola

Persona(Cognome, DataN, Indirizzo)

(1) Accorpamento di ER-relazione derivante da generalizzazione non completa: esempio



Vincolo di generalizzazione:

nessuna istanza di Persona partecipa sia a ISA-S-P sia a ISA-D-P

Il vincolo di generalizzazione diventa sullo schema logico:

Studente[CodiceFiscale] $\cap$ Docente[CodiceFiscale] = $\emptyset$

Persona(<u>CodiceFiscale</u>, Età)

Studente(<u>CodiceFiscale</u>, Matricola)

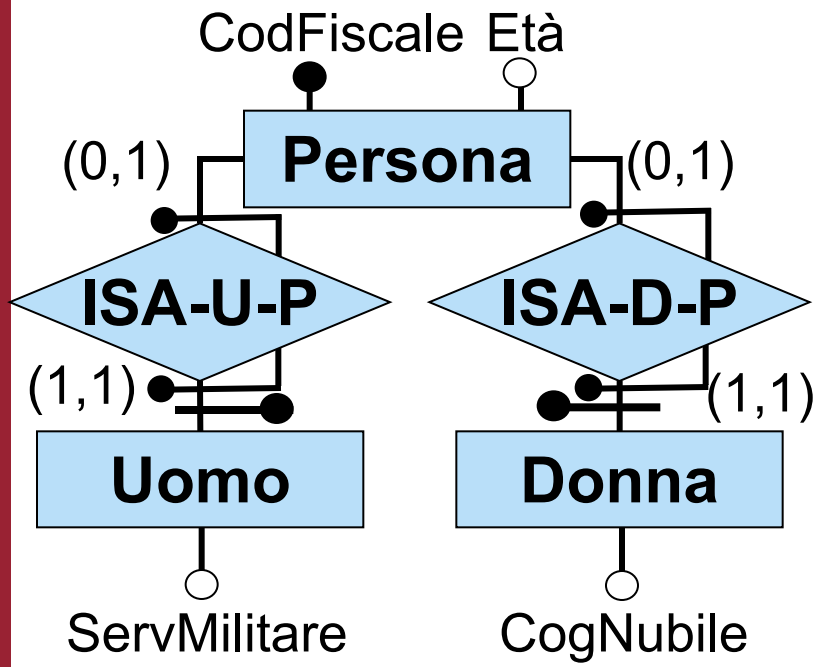
foreign key: Studente[CodiceFiscale] $\subseteq$ Persona[CodiceFiscale]

chiave: Matricola

Docente(<u>CodiceFiscale</u>, Fascia)

foreign key: Docente[CodiceFiscale] $\subseteq$ Persona[CodiceFiscale]

(1) Accorpamento di ER-relazione derivante da generalizzazione completa: esempio



Vincolo di generalizzazione:

ogni istanza di Persona partecipa ad ISA-U-P oppure ad ISA-D-P, ma non ad entrambi

Il vincolo di generalizzazione diventa sullo schema logico:

$Uomo[CodFiscale] \cap Donna[CodFiscale] = \emptyset$

$Persona[CodFiscale] \subseteq Uomo[CodFiscale] \cup Donna[CodFiscale]$

Persona(<u>CodFiscale</u>, Età)

Uomo(<u>CodFiscale</u>, ServMilitare)

foreign key: $Uomo[CodFiscale] \subseteq Persona[CodFiscale]$

Donna(<u>CodFIscale</u>, CogNubile)

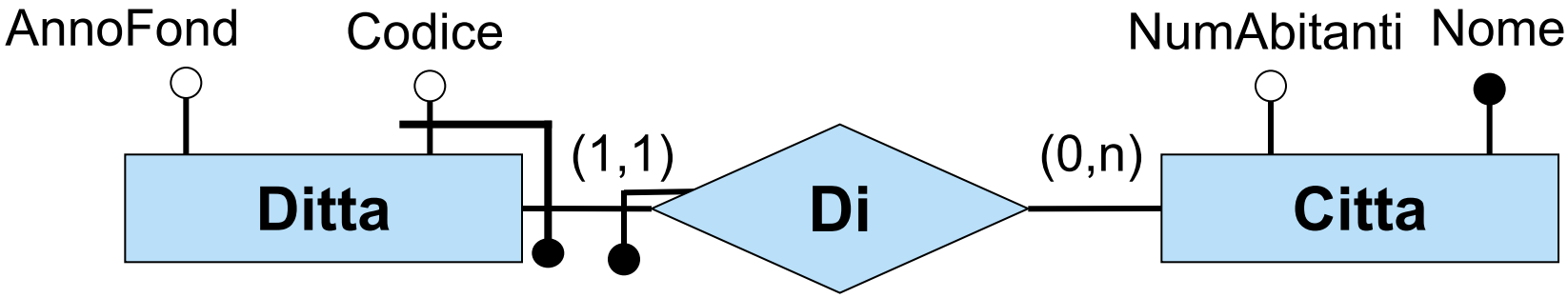
foreign key: $Donna[CodFiscale] \subseteq Persona[CodFiscale]$

(2) Traduzione di ER-relazioni: regole generali

- Ogni ER-relazione **Q** dello schema ER che nel passo precedente è stata accorpata ad una entità **E** non avrà una "sua" relazione nello schema: infatti la relazione corrispondente a **Q** sarà esattamente la relazione R_E che è stata definita per l'entità **E**.
- In altre parole, la relazione R_E rappresenta sia **E**, sia **Q**. Qualunque sia l'istanza I dello schema concettuale, da ogni istanza di **E** e dalla istanza di **Q** ad essa legata si può risalire alla tupla di R_E che le rappresenta entrambe nella base di dati B corrispondente ad I. E, al contrario, qualunque sia la base di dati B dello schema logico, da ogni tupla di R_E in B si può risalire sia alla istanza corrispondente di **E** sia alla istanza corrispondente di **Q** nella istanza I dello schema concettuale corrispondente a B.

(2) Traduzione di ER-relazioni: regole generali

Ad esempio:



Traduzione diretta:

La relazione Ditta rappresenta sia l'entità Ditta sia la ER-relazione Di

Ditta(Codice, Citta, AnnoFond)

foreign key: Ditta[Citta] $\subseteq$ Città[Nome]

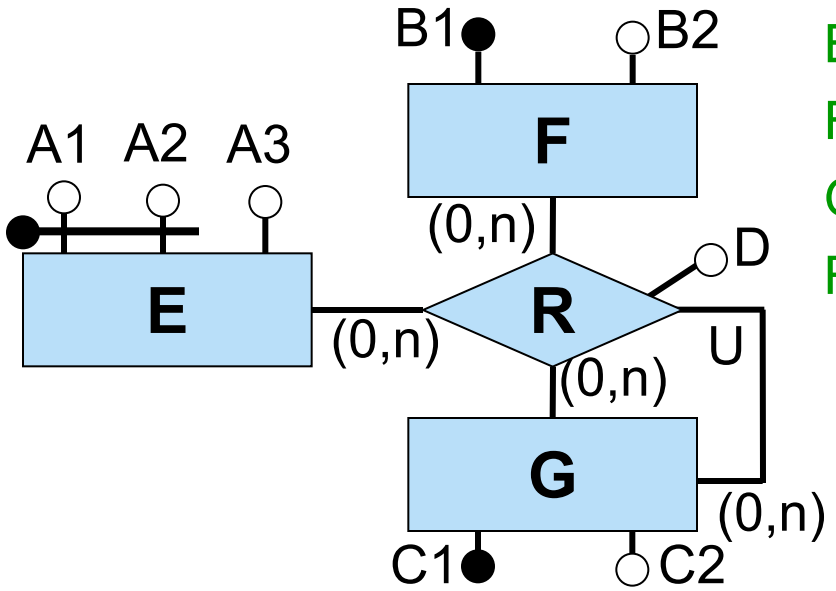
Citta(Nome, NumAbitanti)

(2) Traduzione di ER-relazioni: regole generali

- Ogni ER-relazione **Q** dello schema ER che non è stata accorpata al passo precedente viene tradotta in una relazione R_Q dello schema relazionale.
- Gli **attributi** della relazione R_Q sono:
  - gli attributi della ER-relazione **Q**
  - le chiavi primarie delle entità che partecipano alla ER-relazione **Q;** ciascuna di esse (con tutti gli attributi di cui è formata) viene "replicata" nella relazione R_Q e complessivamente, esse formano la cosiddetta **superchiave implicita** di R_Q
- La scelta della **chiave primaria** di R_Q è basata sull'identificatore principale di R_Q, mentre gli altri identificatori (cioè quelli non principali) essenziali di R_Q danno luogo a **vincoli di chiave** (non primaria) su R_Q
- Le tipizzazioni delle componenti di **Q**, per ogni ruolo, con le entità partecipanti divengono in R_Q vincoli di foreign key (detti **foreign key di ruolo)** verso le relazioni che corrispondono alle entità partecipanti
- A seconda dei casi, possono essere necessari ulteriori vincoli, in particolare, vincoli di foreign key o di inclusione per rappresentare la partecipazione obbligatoria delle entità alla relazione o relazioni ISA con altre ER-relazioni

(2) Traduzione di ER-relazione – caso 1

- Consideriamo il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui l'unico vincolo di identificazione sia quello implicito**.
- La ER-relazione si traduce in una relazione
- Gli **attributi** della relazione sono quelli della ER-relazione, più le chiavi primarie delle relazioni corrispondenti alle entità partecipanti (per ogni ruolo), che saranno oggetto delle foreign key di ruolo
- La **chiave primaria** è determinata dall'identificatore principale della ER-relazione. Ovviamente, poiché l'identificatore principale è quello implicito, la chiave primaria della relazione sarà data dalla superchiave implicita (che in questo caso è chiave)
- Si definiscono i vincoli di foreign key di ruolo, ossia le foreign key dalla relazione verso le entità partecipanti, che realizzano la tipizzazione di **R** nello schema ER



E(A1, A2, A3)

F(B1, B2)

G(C1, C2)

R(EA1, EA2, F, G, U, D)

foreign key: R[EA1,EA2] $\subseteq$ E[A1,A2]

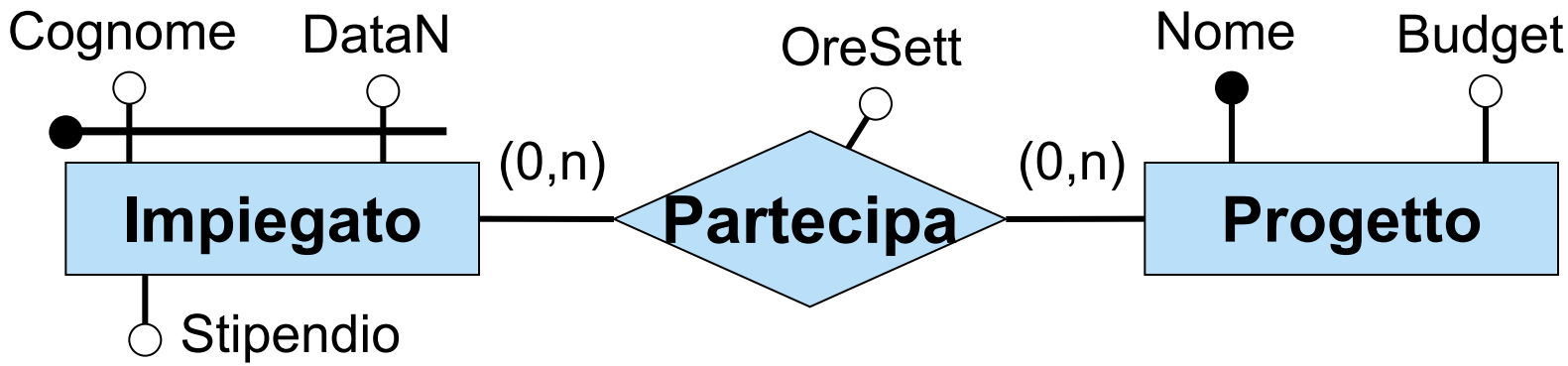
foreign key: R[F] $\subseteq$ F[B1]

foreign key: R[G] $\subseteq$ G[C1]

foreign key: R[U] $\subseteq$ G[C1]

(2) Traduzione di ER-relazione – caso 1: esempio

Nello scegliere per una relazione il nome di un attributo che rappresenta la chiave primaria di un'entità che partecipa alla relazione, può essere opportuno utilizzare il nome del ruolo con cui l'entità partecipa alla relazione (invece del nome che l'attributo ha per l'entità).



Impiegato(Cognome, DataN, Stipendio)

Progetto(Nome, Budget)

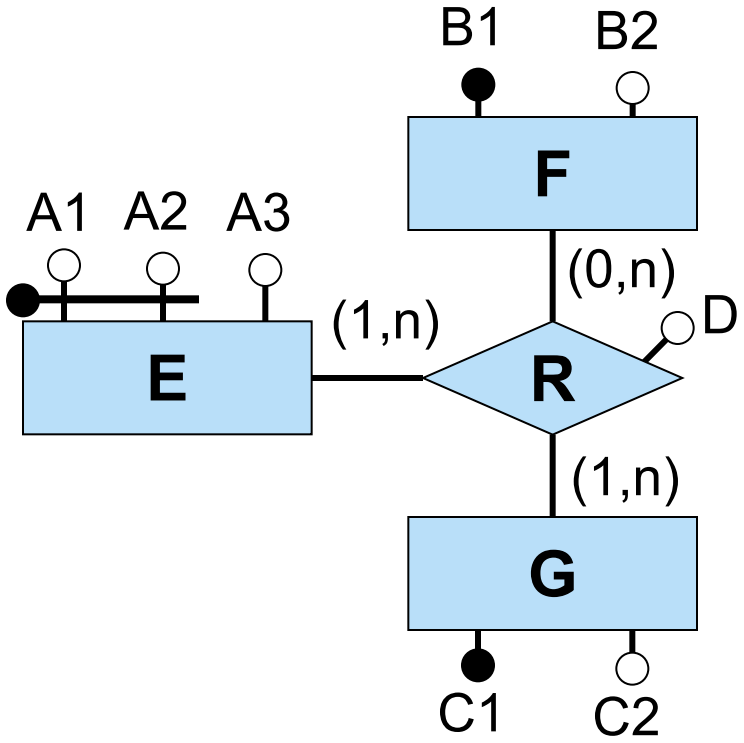
Partecipa(Cognome, DataN, Progetto, OreSett)

foreign key: Partecipa[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]

foreign key: Partecipa[Progetto] $\subseteq$ Progetto[Nome]

(2) Traduzione di ER-relazione – caso 1 con cardinalità minima pari ad 1

- Consideriamo ancora il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui l'unico vincolo di identificazione sia quello implicito**.
- Un vincolo di cardinalità minima 1 per la partecipazione di un'entità (in un ruolo) alla relazione si traduce in un **vincolo di inclusione** dall'entità verso l'attributo (o gli attributi) corrispondenti a quel ruolo nella relazione.
- Il vincolo di inclusione non è in generale di foreign key (si vedano in seguito i casi in cui l'inclusione diventa in realtà una foreign key).



E(A1, A2, A3)

inclusione: E[A1,A2] $\subseteq$ R[A1,A2]

F(B1, B2)

G(C1, C2)

inclusione: G[C1] $\subseteq$ R[C1]

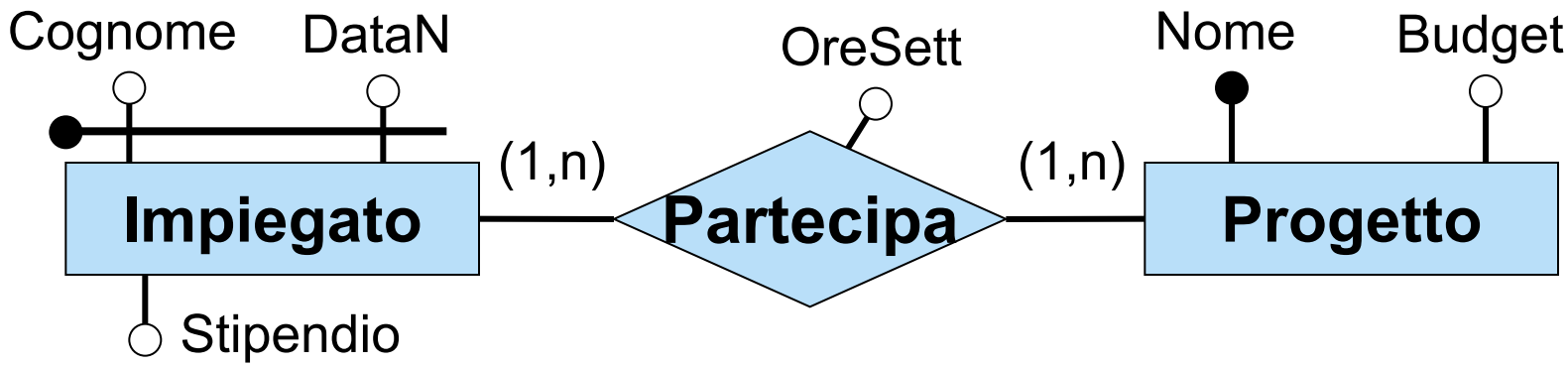
R(A1, A2, B1, C1, D)

foreign key: R[A1,A2] $\subseteq$ E[A1,A2]

foreign key: R[B1] $\subseteq$ F[B1]

foreign key: R[C1] $\subseteq$ G[C1]

(2) Traduzione di ER-relazione – caso 1 con cardinalità minima pari ad 1: esempio



Impiegato(Cognome, DataN, Stipendio)

inclusione: Impiegato[Cognome,DataN] $\subseteq$ Partecipa[Cognome,DataN]

Progetto(Nome, Budget)

inclusione: Progetto[Nome] $\subseteq$ Partecipa[Progetto]

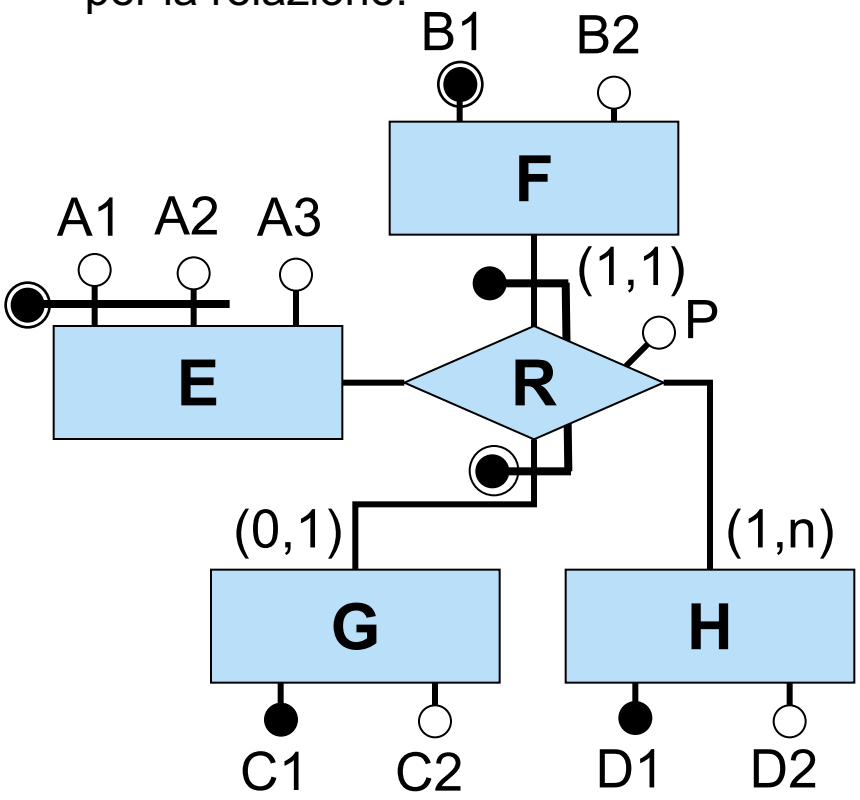
Partecipa(Cognome, DataN, Progetto, OreSett)

foreign key: Partecipa[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]

foreign key: Partecipa[Progetto] $\subseteq$ Progetto[Nome]

(2) Traduzione di ER-relazione – caso 2

- Consideriamo il caso di ER-relazione **R** che **non è stata accorpata ad alcuna entità, ed in cui il vincolo di identificazione principale non sia quello implicito**.
- L'identificatore principale della ER-relazione determina la chiave primaria della relazione. Se l'identificatore principale della relazione è costituito da una sola entità e se tale l'entità ha anche cardinalità minima 1, il vincolo di inclusione corrispondente diventa in realtà un vincolo di foreign key.
- Gli identificatori di relazione che non sono principali si traducono in vincoli di chiave per la relazione.



E(A1, A2, A3)

F(B1, B2)

foreign key: F[B1] $\subseteq$ R[F]

G(C1, C2)

H(D1, D2)

inclusione: H[D1] $\subseteq$ R[H]

R(EA1, EA2, F, G, H, P)

foreign key: R[EA1,EA2] $\subseteq$ E[A1,A2]

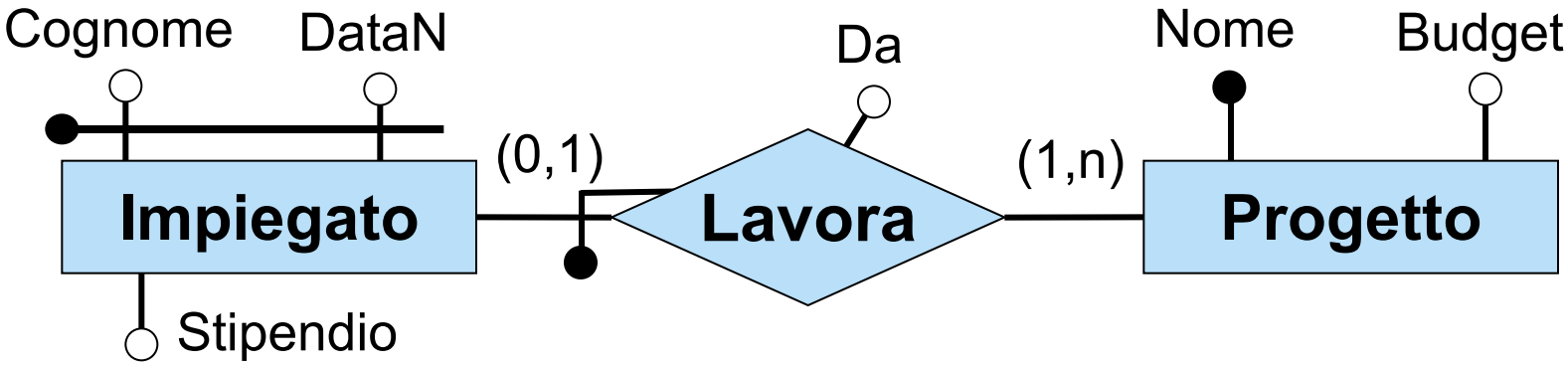
foreign key: R[F] $\subseteq$ F[B1]

foreign key: R[G] $\subseteq$ G[C1]

foreign key: R[H] $\subseteq$ H[D1]

chiave: F

(2) Traduzione di ER-relazione – caso 2: esempio



Impiegato(Cognome, DataN, Stipendio)

Progetto(Nome, Budget)

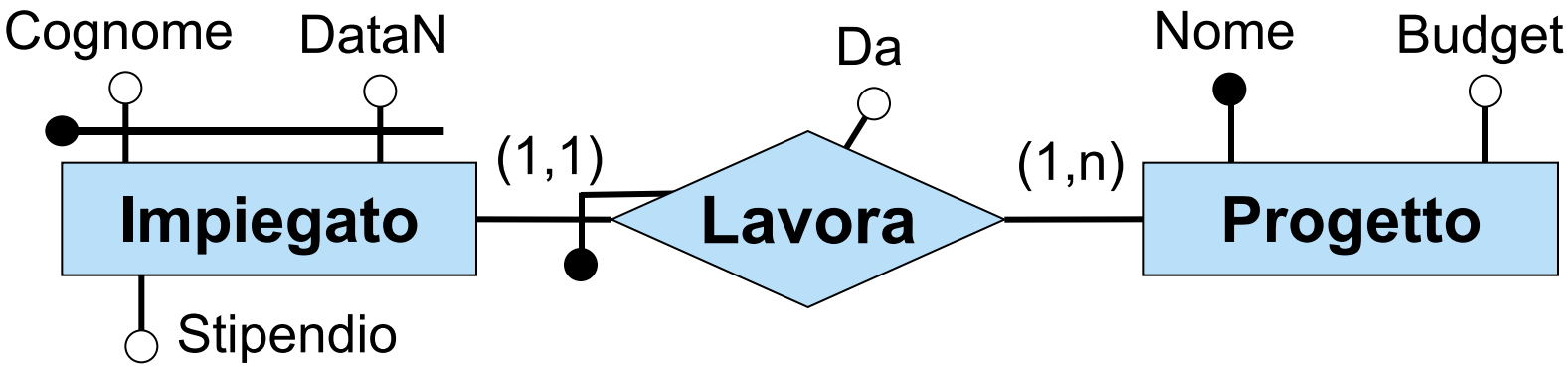
inclusione: Progetto[Nome] $\subseteq$ Lavora[Progetto]

Lavora(Cognome, DataN, Progetto, Da)

foreign key: Lavora[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]

foreign key: Lavora[Progetto] $\subseteq$ Progetto[Nome]

(2) Traduzione di ER-relazione – caso 2 con cardinalità minima pari ad 1: esempio



Impiegato(Cognome, DataN, Stipendio)

foreign key: Impiegato[Cognome,DataN] $\subseteq$ Lavora[Cognome,DataN]

Progetto(Nome, Budget)

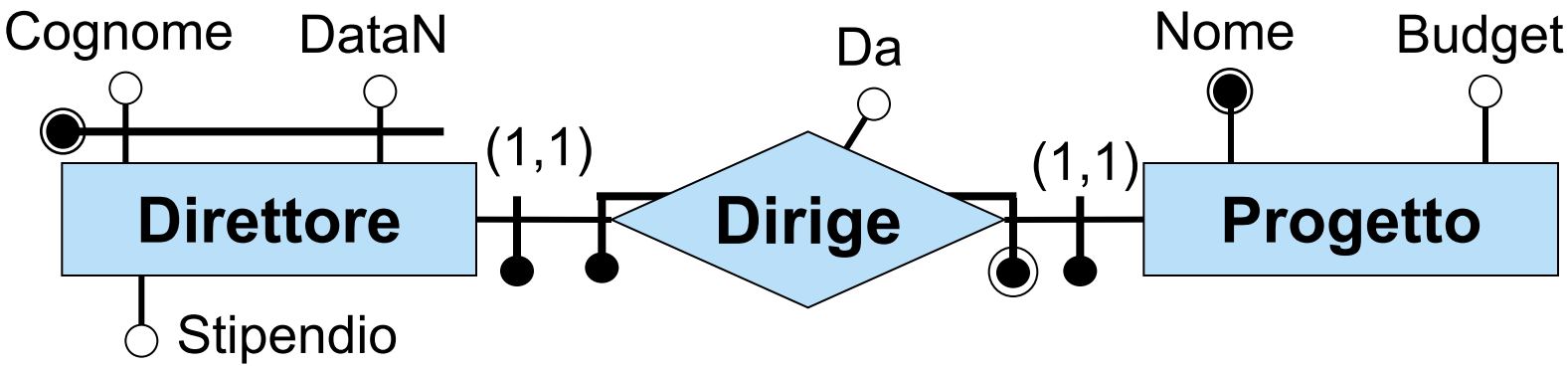
inclusione: Progetto[Nome] $\subseteq$ Lavora[Progetto]

Lavora(Cognome, DataN, Progetto, Da)

foreign key: Lavora[Cognome,DataN] $\subseteq$ Impiegato[Cognome,DataN]

foreign key: Lavora[Progetto] $\subseteq$ Progetto[Nome]

(2) Traduzione di ER-relazione – caso 2: altro esempio



Direttore(Cognome, DataN, Stipendio)

foreign key: Direttore[Cognome,DataN] $\subseteq$ Dirige[Cognome,DataN]

Progetto(Nome, Budget)

foreign key: Progetto[Nome] $\subseteq$ Dirige[Progetto]

Dirige(Cognome, DataN, Progetto, Da)

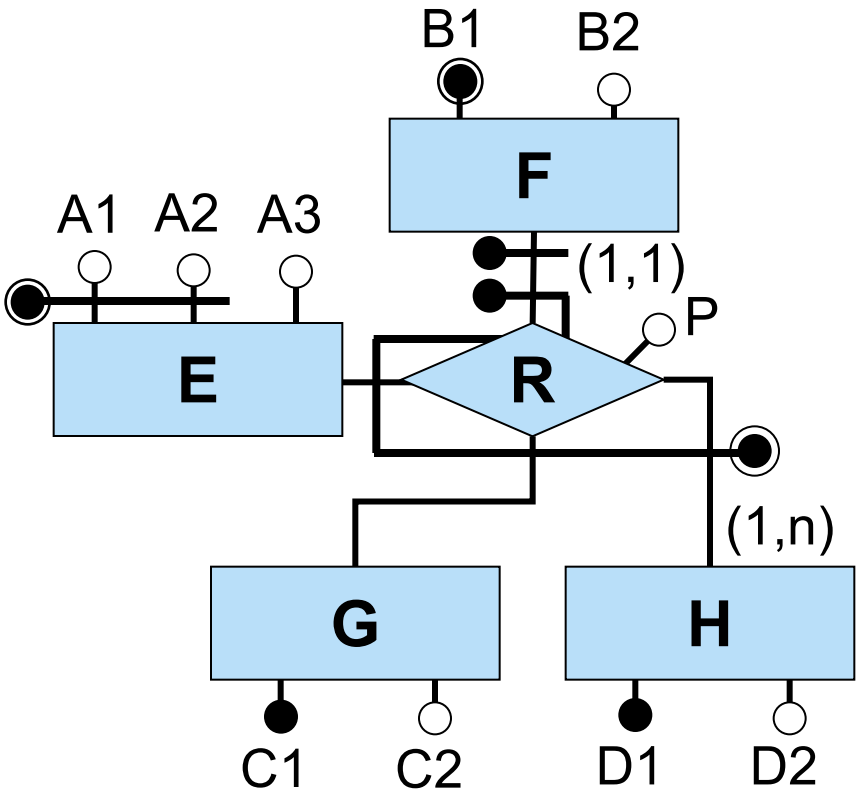
chiave: Cognome, DataN

foreign key: Dirige[Cognome,DataN] $\subseteq$ Direttore[Cognome,DataN]

foreign key: Dirige[Progetto] $\subseteq$ Progetto[Nome]

(2) Traduzione di ER-relazione – caso 2

Il seguente è un altro esempio di caso 2: questa volta il ruolo F della ER-relazione R è identificatore dell'entità F, che equivale alla condizione che tutti i ruoli di R diversi da F formano un identificatore derivato per R. In questo caso quest'ultimo identificatore è stato scelto come identificatore principale di R. Illustriamo come si traduce la relazione R:



E(<u>A1</u>, <u>A2</u>, A3)

F(<u>B1</u>, B2)

foreign key: F[B1] $\subseteq$ R[F]

G(<u>C1</u>, C2)

H(<u>D1</u>, D2)

inclusione: H[D1] $\subseteq$ R[H]

R(<u>EA1</u>, <u>EA2</u>, F, <u>G</u>, <u>H</u>, P)

foreign key: R[EA1,EA2] $\subseteq$ E[A1,A2]

foreign key: R[F] $\subseteq$ F[B1]

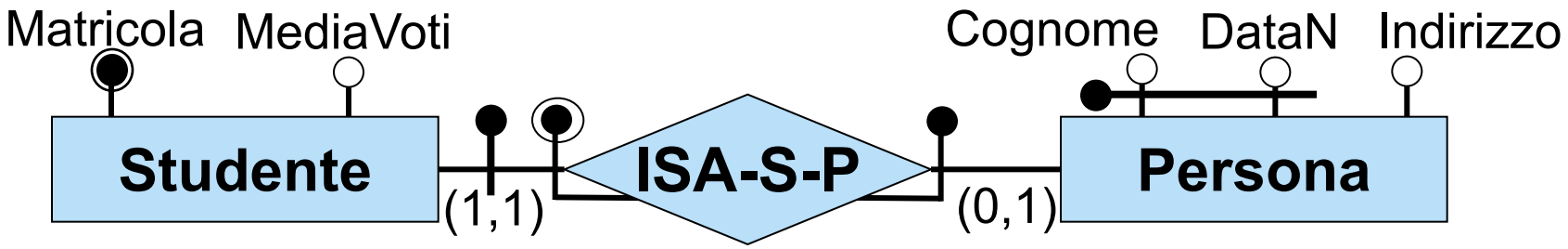
foreign key: R[G] $\subseteq$ G[C1]

foreign key: R[H] $\subseteq$ H[D1]

chiave: F

(2) Traduzione di ER-relazione derivante da ISA: esempio

Una ER-relazione derivante da ISA che non è stata accorpata (perché l'identificatore principale dell'entità che era figlia nello schema ER originario non è quello esterno) si traduce tenendo conto dell'identificatore principale della relazione (che in questo caso sappiamo deve essere indicato). Consideriamo il seguente esempio:



ISA-S-P(Matricola, Cognome, DataN)

foreign key: ISA-S-P[Cognome,DataN] $\subseteq$ Persona[Cognome,DataN]

foreign key: ISA-S-P[Matricola] $\subseteq$ Studente[Matricola]

chiave: Cognome, DataN

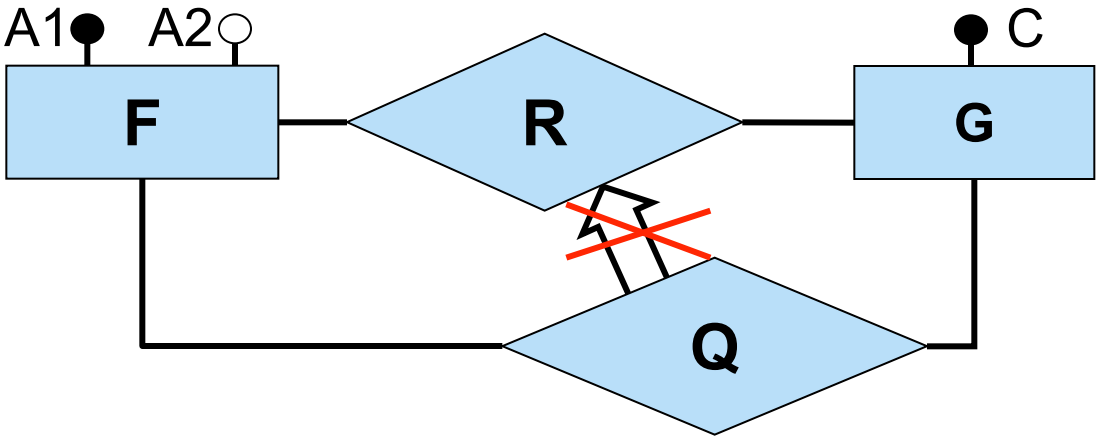
Studente(Matricola, MediaVoti)

foreign key: Studente[Matricola] $\subseteq$ ISA-S-P[Matricola]

Persona(Cognome, DataN, Indirizzo)

(2) Traduzione di vincoli derivanti da ISA tra ER-relazioni

Si ricordi che la ristrutturazione di una ISA tra relazioni ha prodotto un vincolo esterno.



Vincolo esterno:

Ogni istanza di Q è anche istanza di R.

Traduzione: in questo caso il vincolo esterno diventa un vincolo di **foreign key oppure di inclusione** tra la superchiave implicita di Q e la superchiave implicita di R

F(A1, A2)

G(C)

R(F, G)

foreign key: R[F] $\subseteq$ F[A1]

foreign key: R[G] $\subseteq$ G[C]

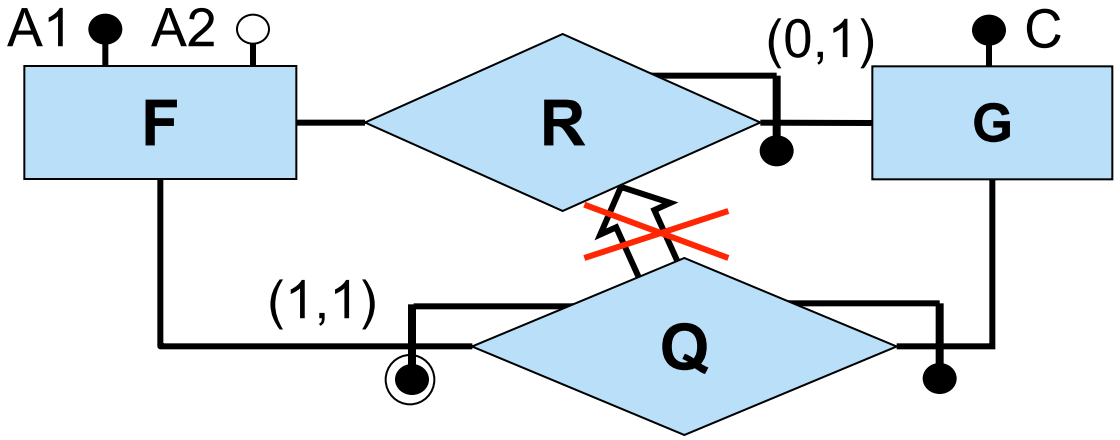
Q(F, G)

foreign key: Q[F,G] $\subseteq$ R[F,G]

Si noti che <F,G> è la superchiave implicita sia di Q sia di R

(2) Traduzione di vincoli derivanti da ISA tra ER-relazioni

Un altro caso:



Vincolo esterno:

Ogni istanza di Q è anche istanza di R.

Traduzione: in questo caso il vincolo esterno diventa un vincolo di **inclusione** tra la superchiave implicita di Q e la superchiave implicita di R

F(A1, A2)

G(C)

R(F, G)

foreign key: $R[F] \subseteq F[A1]$

foreign key: $R[G] \subseteq G[C]$

Q(F, G)

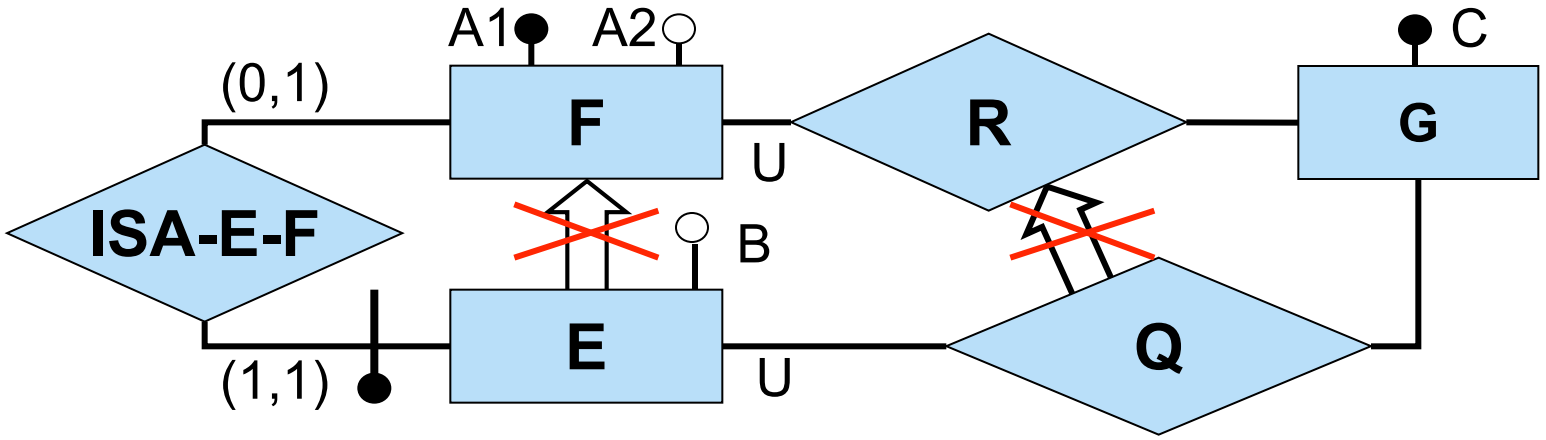
inclusione: $Q[F,G] \subseteq R[F,G]$

chiave: G

Si noti che <F,G> è la superchiave implicita sia di Q sia di R

(2) Traduzione di vincoli derivanti da ISA tra ER-relazioni

Un ulteriore caso:



Vincolo esterno: per ogni istanza (*e*,*g*) di Q, sia *f* l'istanza di F tale che (*e*,*f*) è un'istanza di ISA-E-F (si noti che *f* esiste sempre ed è unica). Allora (*f*,*g*) deve essere un'istanza di R.

Traduzione: il vincolo esterno diventa un vincolo di **foreign key**

E(A1, B) foreign key: E[A1] $\subseteq$ F[A1]

F(A1, A2)

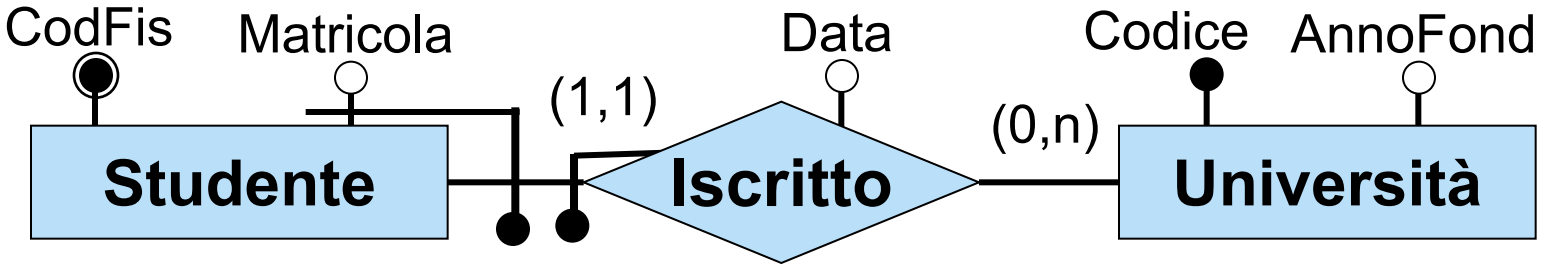
G(C)

R(U, G) foreign key: R[U] $\subseteq$ F[A1] foreign key: R[G] $\subseteq$ G[C]

Q(U, G) foreign key: Q[U] $\subseteq$ E[A1] foreign key: Q[U,G] $\subseteq$ R[U,C]

Torniamo brevemente a (1): traduzione di entità con identificatore esterno non principale

Ora che sappiamo come si traducono le relazioni, possiamo tornare brevemente alla traduzione delle entità per osservare che un vincolo di identificazione non principale esterna per una entità E diventa un vincolo esterno (cosiddetto "di join") che riguarda sia l'entita E sia la relazione che partecipa all'identificazione.



Iscritto(Studente, Università, Data)

foreign key: Iscritto[Studente] $\subseteq$ Studente[CodFis]

foreign key: Iscritto[Università] $\subseteq$ Università [Codice]

Studente(CodFis, Matricola)

foreign key: Studente[CodFis] $\subseteq$ Iscritto[Studente]

Università(Codice, AnnoFond)

Vincolo: nell'equi-join tra Studente e Iscritto sugli attributi CodFis e Studente, gli attributi Matricola e Università formano una chiave

(3) Traduzione di vincoli

Questi sono i vincoli più rilevanti da considerare nella traduzione:

- Vincoli **not null** per gli attributi obbligatori
- Vincoli di interdipendenza di valori nulli (provenienti da attributi composti opzionali), formulati come vincoli di tupla
- Vincoli di **chiave** (primarie e non)
- Vincoli di **foreign key** che provengono
  - dalla tipizzazione di ER-relazioni (incluse quelle che sono state accorpate in entità)
  - dai vincoli esterni derivanti dall'ISA di ER-relazioni
- Vincoli **di generalizzazione**, formulati come vincoli insiemistici
- Vincoli di cardinalità:
  - La partecipazione obbligatoria (cardinalità minima 1) diventa **vincolo di inclusione** o **foreign key** dalla relazione che corrisponde all'entità a quella che corrisponde alla ER-relazione
  - La funzionalità (cardinalità massima 1) diventa **vincolo di chiave** sulla relazione che corrisponde alla ER-relazione
  - Gli altri vincoli di cardinalità diventano vincoli esterni
- Gli altri vincoli esterni vanno opportunamente tradotti

(4) Riformulazione di operazioni e carico applicativo

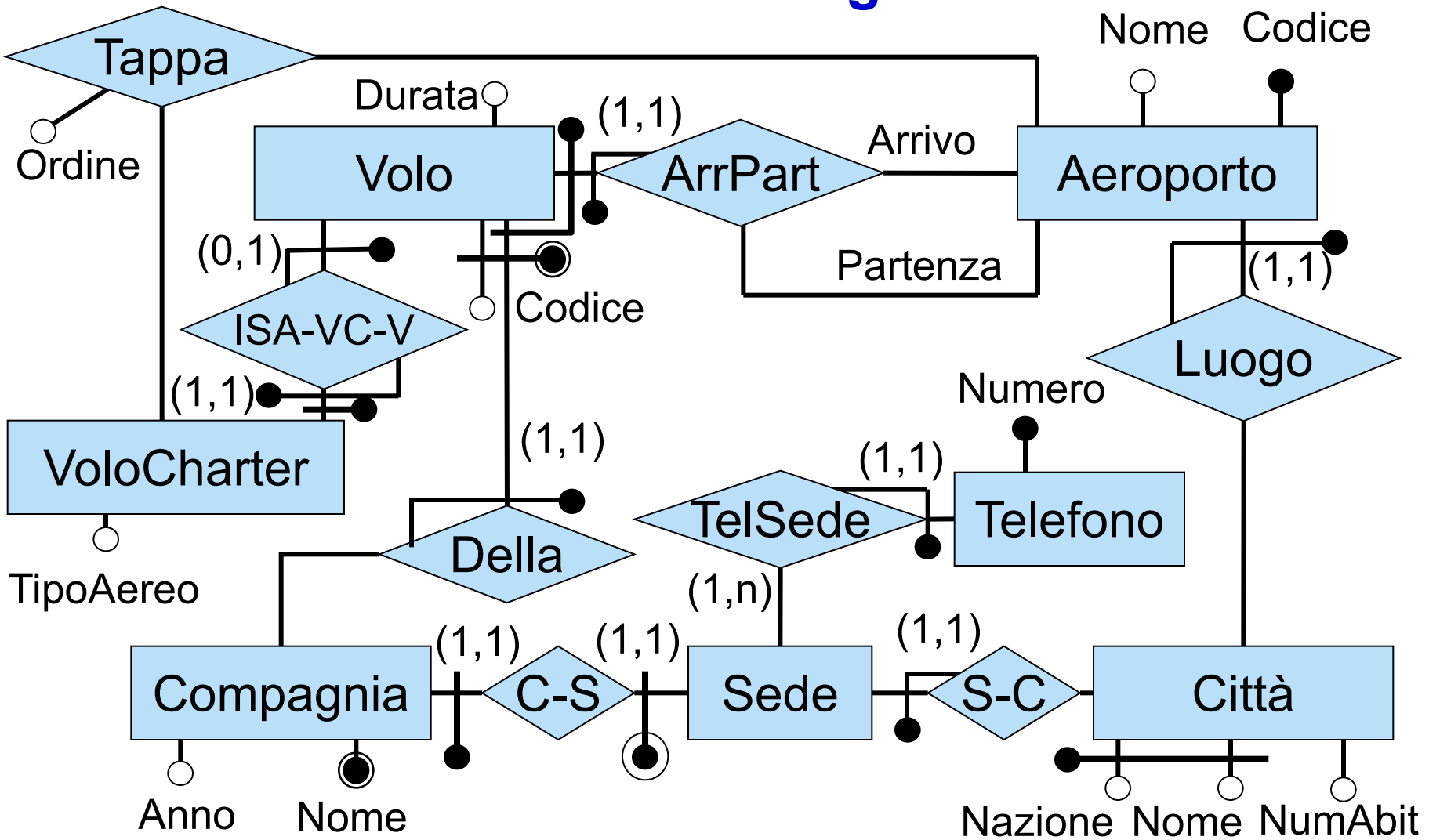
- Le operazioni e le informazioni sul carico applicativo sono state espresse all'inizio della progettazione logica sulla base dello schema concettuale, e poi modificate per renderle coerenti con lo schema concettuale ristrutturato. È ora necessario riformulare le operazioni e le informazioni sul carico applicativo in modo che siano coerenti con lo schema logico

- La riformulazione viene condotta semplicemente tenendo presente come le entità e le relazioni dello schema Entità-Relazione ristrutturato sono state tradotte nello schema relazionale

Riassunto sulla traduzione diretta

1. Traduzione di ogni entità in una relazione, con i seguenti attributi:
   - gli attributi dell'entità stessa
   - gli attributi delle relazioni che partecipano all'identificazione principale esterna dell'entità, insieme alle chiavi primarie, opportunamente nominate (possibilmente con ruolo), delle entità connesse a tali relazioni (si noti che in questo caso, per l'assenza di cicli sull'identificazione principale esterna, la relazione non può avere altre entità per le quali la relazione è parte di identificatore esterno)
2. Traduzione di ogni ER-relazione (non accorpata al punto 1) in relazione, con opportuna chiave primaria, e con i seguenti attributi:
   - gli identificatori principali delle entità partecipanti (con opportuno nome)
   - gli attributi della ER-relazione
3. Traduzione di vincoli
   - not null per gli attributi obbligatori
   - chiavi (primarie e non)
   - foreign key o vincoli di inclusione che provengono dall'accorpamento (vedi punto 1), da tipizzazione di relazioni, da ISA di relazioni
   - vincoli di generalizzazione
   - vincoli di cardinalità (parte obbligatoria diventa vincolo di inclusione, parte di funzionalità diventa vincolo di chiave)
   - altri vincoli esterni (in particolare i vincoli di join da entità con identificatore esterno non principale)
4. Riformulazione di operazioni e specifiche sul carico applicativo in termini dello schema logico

Esercizio 5: tradurre il seguente schema



Vincolo esterno: per ogni v in VoloCharter, se $(v,a_1),...,(v,a_n)$ sono tutte le coppie in Tappa alle quali partecipa v, e se $o_1,...,o_n$ sono i valori assegnati a tali coppie dall'attributo Ordine, allora per i=1,...,n esiste un o_j tale che o_j=i.

Esercizio 5: soluzione (parte 1)

Volo(Codice, Comp, Durata)
- foreign key: Volo[Comp] $\subseteq$ Compagnia[Nome]
- foreign key: Volo[Codice,Comp] $\subseteq$ ArrPart[Codice,Comp]

ArrPart(Codice, Comp, Arrivo, Partenza)
- foreign key: ArrPart[Arrivo] $\subseteq$ Aeroporto[Codice]
- foreign key: ArrPart[Partenza] $\subseteq$ Aeroporto[Codice]
- foreign key: ArrPart[Codice,Comp] $\subseteq$ Volo[Codice,Comp]
- chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo)
- foreign key: VoloCharter[Codice,Comp] $\subseteq$ Volo[Codice,Comp]

Aeroporto(Codice, Nome)
- foreign key: Aeroporto[Codice] $\subseteq$ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
- foreign key: LuogoAeroporto[Aeroporto] $\subseteq$ Aeroporto[Codice]
- foreign key: LuogoAeroporto[NomeCittà,NazCittà] $\subseteq$ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

Nota: viene dall'identificatore esterno non principale di Volo. In questo caso non è necessario ricorrere ad un vincolo sul join perché tutti gli attributi rilevanti si trovano nella relazione ArrPart

Esercizio 5: soluzione (parte 2)

Compagnia(Nome, AnnoFond)
  foreign key: Compagnia[Nome] $\subseteq$ Sede[Comp]

Sede(Comp)
  foreign key: Sede[Comp] $\subseteq$ Compagnia[Nome]
  foreign key: Sede[Comp] $\subseteq$ CittàSede[Comp]
  inclusione: Sede[Comp] $\subseteq$ TelefonoComp[Comp]

CittàSede(Comp, NomeCittà, NazCittà)
  foreign key: CittàSede[Comp] $\subseteq$ Sede[Comp]
  foreign key: CittàSede[NomeCittà,NazCittà] $\subseteq$ Città[Nome,Nazione]

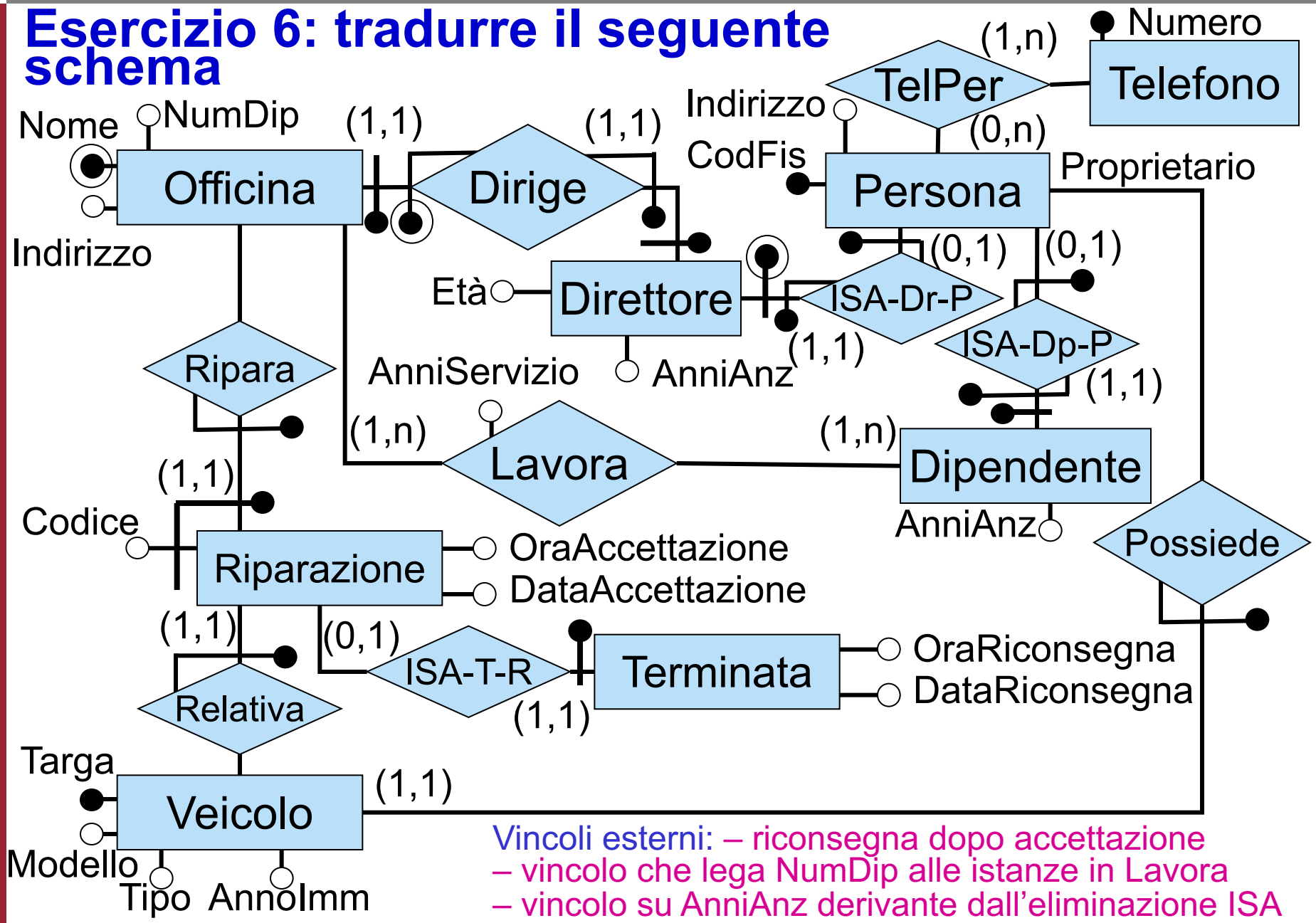
TelefonoSede(Numero, Sede)
  foreign key: TelefonoSede[Sede] $\subseteq$ Sede[Comp]
  foreign key: TelefonoSede[Numero] $\subseteq$ Telefono[Numero]

Telefono(Numero)
  foreign key: Telefono[Numero] $\subseteq$ TelefonoComp[Numero]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
  foreign key: Tappa[CodVoloCharter,Comp] $\subseteq$ VoloCharter[Codice,Comp]
  foreign key: Tappa[Aeroporto] $\subseteq$ Aeroporto[Codice]

Vincolo esterno: per ogni v in Tappa[CodVoloCharter], se $o_1,...,o_n$ sono i valori che compaiono nell'attributo Ordine nelle tuple di Tappa che hanno v come valore nell'attributo CodVoloCharter, allora per i=1,...,n esiste uno ed un solo o_j tale che o_j=i.

Esercizio 6: tradurre il seguente schema



Vincoli esterni: – riconsegna dopo accettazione
– vincolo che lega NumDip alle istanze in Lavora
– vincolo su AnniAnz derivante dall'eliminazione ISA

Esercizio 6: soluzione (parte 1)

Officina(Nome, NumDip, Indirizzo)
- foreign key: Officina[Nome] $\subseteq$ Dirige[Officina]
- inclusione: Officina[Nome] $\subseteq$ Lavora[Officina]

Persona(CodFis, Indirizzo)

Direttore(CodFis, Età, AnniAnz)
- foreign key: Direttore[CodFis] $\subseteq$ Persona[CodFis]
- foreign key: Direttore[CodFis] $\subseteq$ Dirige[Direttore]

Dipendente(CodFis, AnniAnz)
- foreign key: Dipendente[CodFis] $\subseteq$ Persona[CodFis]
- inclusione: Dipendente[CodFis] $\subseteq$ Lavora[Dipendente]

Dirige(Officina, Direttore)
- foreign key: Dirige[Officina] $\subseteq$ Officina[Nome]
- foreign key: Dirige[Direttore] $\subseteq$ Direttore[CodFis]
- chiave: Direttore

Lavora(Officina, Dipendente, AnniServizio)
- foreign key: Lavora[Officina] $\subseteq$ Officina[Nome]
- foreign key: Lavora[Dipendente] $\subseteq$ Dipendente[CodFis]

TelPer(CodFis, Telefono)
- foreign key: TelPer[CodFis] $\subseteq$ Persona[CodFis]
- foreign key: TelPer[Telefono] $\subseteq$ Telefono[Numero]

Esercizio 6: soluzione (parte 2)

Telefono(Numero)
  inclusione: Telefono[Numero] $\subseteq$ TelPer[Telefono]

Veicolo(Targa, Modello, Tipo, AnnoImm)
  foreign key : Veicolo[Targa] $\subseteq$ Possiede[Veicolo]

Possiede(Veicolo, Proprietario)
  foreign key: Possiede[Veicolo] $\subseteq$ Veicolo[Targa]
  foreign key: Possiede[Proprietario] $\subseteq$ Persona[CodFis]

Riparazione(Codice, Officina, OraAcc, DataAcc)
  foreign key: Riparazione[Officina] $\subseteq$ Officina[Nome]
  foreign key: Riparazione[Codice,Officina] $\subseteq$ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
  foreign key: Relativa[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]
  foreign key: Relativa[Veicolo] $\subseteq$ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
  foreign key: Terminata[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]

Vincoli esterni:

- riconsegna dopo accettazione
- vincolo che lega Officina[NumDip] alle istanze in Lavora
- vincolo su AnniAnz di Direttore e Dipendente derivante dall'eliminazione ISA

6. La progettazione logica

6.4 ristrutturazione dello schema logico

1. introduzione alla progettazione logica
2. ristrutturazione dello schema ER
3. traduzione diretta nel modello relazionale
4. **ristrutturazione dello schema logico**

Cosa sappiamo dopo la traduzione

- Abbiamo rispettato la modularizzazione concettuale
- Si possono presentare potenziali problemi di efficienza rispetto allo spazio
  - valori nulli (solo quelli dovuti ad attributi opzionali)
  - ci possono essere due relazioni R_1 e R_2 con chiavi K_1 e K_2 tali che valga sia $R_1[K_1] \subseteq R_2[K_2]$ sia $R_2[K_2] \subseteq R_1[K_1]$
  - ridondanze lasciate
- Si possono presentare potenziali problemi di efficienza rispetto al tempo di esecuzione delle query e degli update
  - ridondanze lasciate (incidono sugli update)
  - numero e struttura delle relazioni
- Ci devono essere dei buoni motivi legati all'**efficienza** per cambiare le scelte fatte. Ma se queste ragioni ci sono, si tratta adesso di ristrutturare lo schema in modo che la base di dati sia equivalente a quella ottenuta dalla traduzione diretta e sia più efficiente rispetto alle operazioni dell'applicazione.

Modello di costo

- Una relazione occupa un certo numero di pagine di memoria secondaria
- Il numero di accessi alle pagine della memoria secondaria domina sull'elaborazione in memoria centrale
- Un accesso a memoria secondaria avviene ad una pagina intera
- Il numero complessivo di pagine accedute allo scopo di effettuare un'operazione dipende in generale:
  - dal tipo di operazione (lettura o scrittura)
  - dal numero di tuple delle relazioni coinvolte (che influisce sul numero di pagine delle relazioni coinvolte: più tuple, più pagine)
  - dal numero di tuple che possono essere memorizzate in una pagina di memoria secondaria (questo numero influisce sul numero di pagine delle relazioni coinvolte: più tuple in una pagina, meno pagine accedute)

Modello di costo: esempi

Per una tabella (o relazione) **R** denotiamo con

- $N_P(R)$: numero di pagine in memoria secondaria occupate da **R**
- $N_{TP}(R)$: numero di tuple per ogni pagina di **R**

Stima di massima del costo delle operazioni, espresso tenendo presente che il costo dominante è quello per accedere alla pagine:

- la selezione su una relazione **R** ha costo pari a $N_P(R)$
- la proiezione su una relazione **R** ha costo pari a $N_P(R)$
- il join di **R** con **Q** si basa su un doppio ciclo
  - se non ci sono indici, allora il costo è $N_P(R) \times N_P(Q)$:

    per ogni pagina **p1** di **R**, per ogni pagina **p2** di **Q**
    per ogni tupla **r** di **R** in **p1**, per ogni tupla **q** di **Q** in **p2**
    se **r** e **q** sono in join, metti la tupla nel risultato
  - se **Q** ha un indice molto selettivo sull'attributo di join, allora il costo è $N_P(R) + N_P(R) \times N_{TP}(R)$

Criteri generali per individuare potenziali problemi

- relazione con
  - tante tuple → la relazione occupa in genere molte pagine
  - tanti attributi → una pagina contiene poche tuple e questo può comportare un grande numero di pagine
- attributi con tanti valori nulli
  → spreco di spazio → una pagina contiene poche tuple
- la proiezione è costosa (quando una pagina contiene poche tuple e specialmente quando si devono eliminare i duplicati)
- il join è costoso (quasi sempre)
- la verifica di vincoli è costosa (quasi sempre)

Conclusione:

per una relazione **R** rilevante (con tante tuple) occorre tentare di tenere basso $N_P(R)$, al limite aumentando $N_{TP}(R)$, perché, a parità di altre condizioni, poche tuple in ogni pagina significa un numero maggiore di pagine. Aumentare $N_{TP}(R)$ può significare spezzare (**decomporre**) le relazioni. D'altra parte, per eliminare la necessità di join costosi potremmo essere indotti ad **accorpare** le relazioni.

Operazioni di ristrutturazione dello schema logico

La ristrutturazione dello schema si effettua eseguendo iterativamente una serie di operazioni sulle relazioni dello schema logico. Considereremo due tipi di operazioni

- **Operazioni di decomposizione**
  - che spezzano una relazione al fine di produrre da essa due relazioni che offrono maggiore possibilità di efficienza rispetto a quella originale
  - e che sono caratterizzate da tre tipi differenti di decomposizione: verticale, orizzontale e mista.
- **Operazioni di accorpamento**
  - che uniscono due relazioni al fine di produrre da esse una relazione che offre maggiore possibilità di efficienza rispetto a quella originale
  - e che sono caratterizzate da tre tipi differenti di accorpamento: per evitare il join, per eliminare relazioni inutile e per eliminare attributi inutili.

In entrambi i casi, le relazioni originarie possono essere ricostruite attraverso la definizione di opportune viste (view).

Nota: le ristrutturazioni si applicano in presenza di determinati attributi che formano una chiave di relazione. Sulle slide indicheremo tali attributi con **K**, intendendo che **K** è una chiave (composta da uno o più attributi) della relazione corrispondente

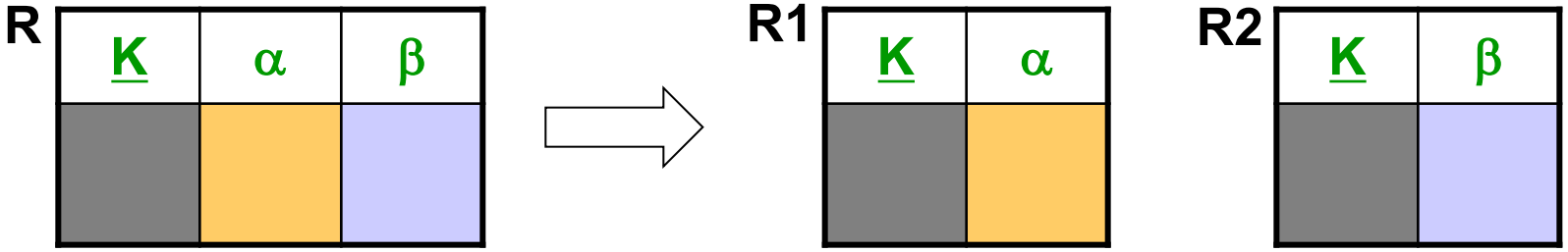
Operazioni di ristrutturazione dello schema logico

Nel seguito ci concentriamo sulle seguenti operazioni:

- **Decomposizione**
  - *Verticale* (sempre sulla chiave)
    - per facilitare l'accesso (con selezioni e proiezioni)
    - per normalizzazione (ignoreremo largamente questo aspetto)
  - *Orizzontale*
    - per facilitare l'accesso (con selezioni)
  - *Mista* (ottenibile come composizione delle altre due)
    - per evitare valori nulli
- **Accorpamento**
  - per facilitare l'accesso (evita join)
  - per eliminare relazioni inutili
  - per eliminare attributi inutili

Nota importante: è bene osservare che potremmo anche considerare ulteriori operazioni, in particolare quelle che introducono ridondanze. Per semplicità, però, non faremo riferimento a tali operazioni.

Decomposizione verticale per facilitare l'accesso



Vincoli dello schema ristrutturato:

- foreign key: $R1[K] \subseteq R2[K]$
  foreign key: $R2[K] \subseteq R1[K]$
- vincoli di inclusione da e verso **R** si riformulano su **R1** (o **R2**)
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando gli accessi ad **R** avvengono prevalentemente in modo separato sugli attributi α rispetto agli attributi β (α e β denotano insiemi di attributi)
- La decomposizione è vantaggiosa perché entrambi i valori N_P**(R1)** e N_P**(R2)** sono più bassi rispetto a N_P**(R)**
- La relazione **R** può essere ricostruita attraverso una vista che calcola il join tra **R1** ed **R2** su **K** (ma ovviamente l'esigenza di ricostruire il join deve essere rara)

Decomposizione verticale: esempio

Immobile(Codice, Particella, Zona, Indirizzo, Città, Mq, NumVani, Valore)

foreign key: Immobile[Città] $\subseteq$ Città [Cod]

Agenzia(CodiceAg, Immobile)

foreign key: Agenzia[Immobile] $\subseteq$ Immobile[Codice]

Supponiamo che ai dati catastali degli immobili (particella, zona, indirizzo, città) si acceda prevalentemente in modo separato rispetto ai dati commerciali (metri quadri, numero di vani, valore). Applichiamo quindi la decomposizione verticale, ed otteniamo:

ImmobileCatasto(Codice, Particella, Zona, Indirizzo, Città)

foreign key: ImmobileCatasto[Città] $\subseteq$ Città [Cod]

foreign key: ImmobileCatasto[Codice] $\subseteq$ ImmobileComm [Codice]

ImmobileComm(Codice, Mq, NumVani, Valore)

foreign key: ImmobileComm[Codice] $\subseteq$ ImmobileCatasto[Codice]

Agenzia(CodiceAg, Immobile)

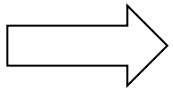
foreign key: Agenzia[Immobile] $\subseteq$ ImmobileCatasto[Codice]

Si noti come la relazione Immobile originaria si possa sempre ricostruire mediante l'equi-join tra ImmobileCatasto e ImmobileComm sull'attributo Codice

Decomposizione orizzontale per facilitare l'accesso

R

K1	α
k11	a11
k12	a12
…..	…..
k21	a21
k22	a22
…..	……



R1

K1	α
k11	a11
k12	a12
…..	…..

R2

K1	α
k21	a21
k22	a22
…..	…..

Se le fasce sono disgiunte e vogliamo essere sicuri che non vi siano errori nell'inserimento di tuple:

$R1[K1] \cap R2[K1] = \varnothing$

Ulteriori vincoli dello schema ristrutturato:

- vincoli di inclusione da **R** diventano vincoli di inclusione da **R1** e da **R2**
- vincoli di inclusione a **R** diventano vincoli di inclusione a **R1** $\cup$ **R2**
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando gli accessi alle tuple di **R** di una delle due "fasce" avvengono separatamente dagli accessi alle tuple dell'altra fascia
- La decomposizione è vantaggiosa perché entrambi i valori $N_P(R1)$ e $N_P(R2)$ sono più bassi rispetto a $N_P(R)$
- **R** può essere ricostruita attraverso una vista che calcola l'unione di **R1** ed **R2**

Decomposizione orizzontale: esempio

Telefonata(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)

foreign key: Telefonata[UtenzaInvio] $\subseteq$ Utenza[Cod]

foreign key: Telefonata[UtenzaDestinazione] $\subseteq$ Utenza[Cod]

Centrale(Codice, Telefonata)

foreign key: Centrale[Telefonata] $\subseteq$ Telefonata[Codice]

Supponiamo che alle telefonate si acceda per fasce orarie (giorno, sera, notte) determinate sul tempo di inizio. Applichiamo due volte in cascata la decomposizione orizzontale (opportuni vincoli detteranno le regole per i valori corretti del campo OrarioInizio delle tabelle e realizzeranno quindi i vincoli di disgiuntezza), ed otteniamo:

TelefonataGiorno(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)

foreign key: TelefonataGiorno[UtenzaInvio] $\subseteq$ Utenza[Cod]

foreign key: TelefonataGiorno[UtenzaDestinazione] $\subseteq$ Utenza[Cod]

TelefonataSera(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)

foreign key: TelefonataSera[UtenzaInvio] $\subseteq$ Utenza[Cod]

foreign key: TelefonataSera[UtenzaDestinazione] $\subseteq$ Utenza[Cod]

TelefonataNotte(Codice, OrarioInizio, OrarioFine, UtenzaInvio, UtenzaDestinazione)

foreign key: TelefonataNotte[UtenzaInvio] $\subseteq$ Utenza[Cod]

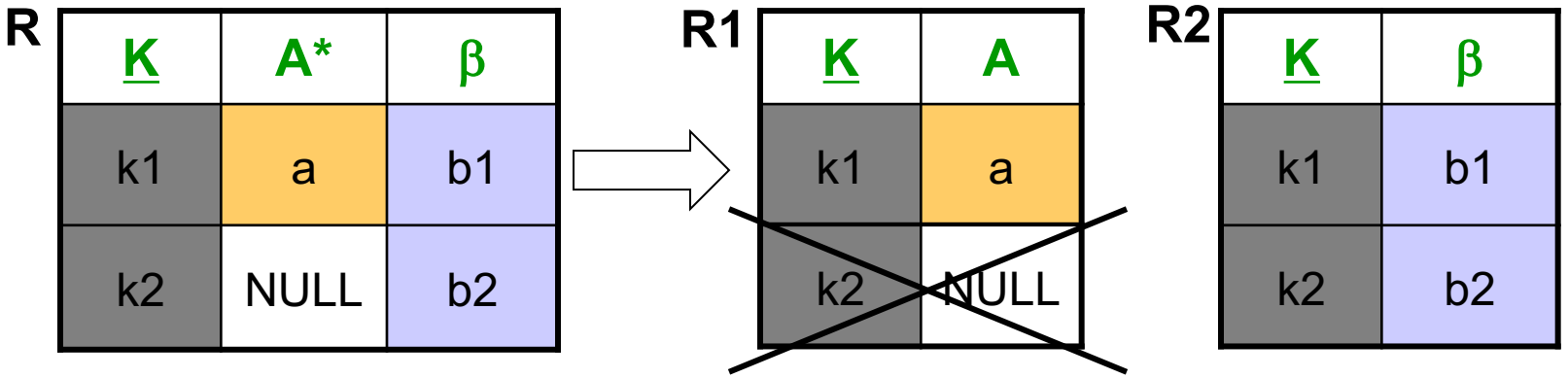
foreign key: TelefonataNotte[UtenzaDestinazione] $\subseteq$ Utenza[Cod]

Centrale(Codice, Telefonata)

Vincolo esterno:

Centrale[Telefonata] $\subseteq$ TelefonataGiorno[Codice] $\cup$ TelefonataSera[Codice] $\cup$ TelefonataNotte[Codice]

Decomposizione mista per evitare valori nulli



Mostriamo un esempio di come si possano combinare i due tipi di decomposizione che abbiamo illustrato. Supponiamo che **A** sia un attributo su cui si vogliono evitare valori nulli, e che β denoti un insieme di attributi. Per eliminare i valori nulli possiamo effettuare una decomposizione verticale seguita da una orizzontale su **R1**, seguita a sua volta dall'eliminazione della parte di **R1** con valori nulli in **A** (che a questo punto diventa inutile). Si noti che avremmo anche potuto prima decomporre orizzontalmente e poi verticalmente; si lascia come esercizio la verifica della forma di schema che si otterrebbe in questo modo, che è comunque accettabile.

Vincoli dello schema ristrutturato:

- foreign key: $R1[K] \subseteq R2[K]$
- vincoli di inclusione da e verso **R** diventano inclusioni da e verso **R2**
- tutti gli altri vincoli che coinvolgono **R** vanno riformulati

- Si applica quando la relazione ha tanti valori nulli in **A** e quindi si rischia di sprecare spazio
- La decomposizione è vantaggiosa perché entrambi i valori $N_P(R1)$ e $N_P(R2)$ sono più bassi rispetto a $N_P(R)$
- **R** può essere ricostruita attraverso una vista che calcola il join esterno tra **R1** ed **R2** su **K**.

Decomposizione mista: esempio

Persona(CodFiscale, CittàNascita, NumTel*, DataMatrimonio*)

foreign key: Persona[CittàNascita] $\subseteq$ Città[Cod]

Città(Cod, Sindaco)

foreign key: Città[Sindaco] $\subseteq$ Persona[CodFiscale]

Supponiamo di non volere valori nulli. Applichiamo quindi due volte in cascata la decomposizione mista, ed otteniamo:

Persona(CodFiscale, CittàNascita)

foreign key: Persona[CittàNascita] $\subseteq$ Città[Cod]

PersonaConTelefono(CodFiscale, NumTel)

foreign key: PersonaConTelefono[CodFiscale] $\subseteq$ Persona[CodFiscale]

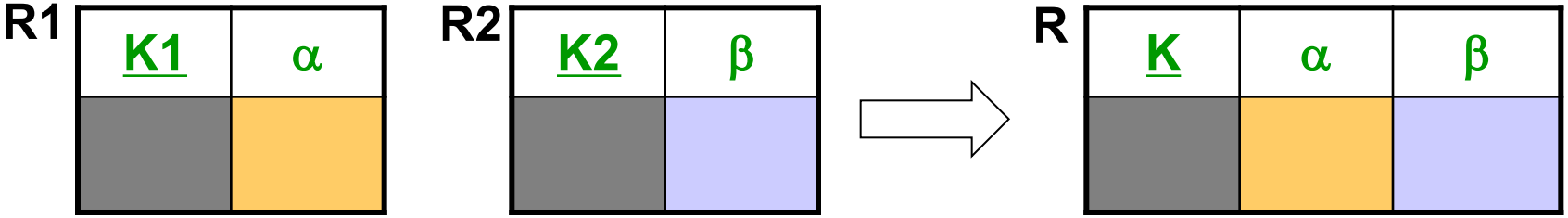
PersonaSposata(CodFiscale, DataMatrimonio)

foreign key: PersonaSposata[CodFiscale] $\subseteq$ Persona[CodFiscale]

Città(Cod, Sindaco)

foreign key: Città[Sindaco] $\subseteq$ Persona[CodFiscale]

Accorpamento per facilitare l'accesso: caso 1



foreign key: R1[K1] ⊆ R2[K2]

foreign key: R2[K2] ⊆ R1[K1]

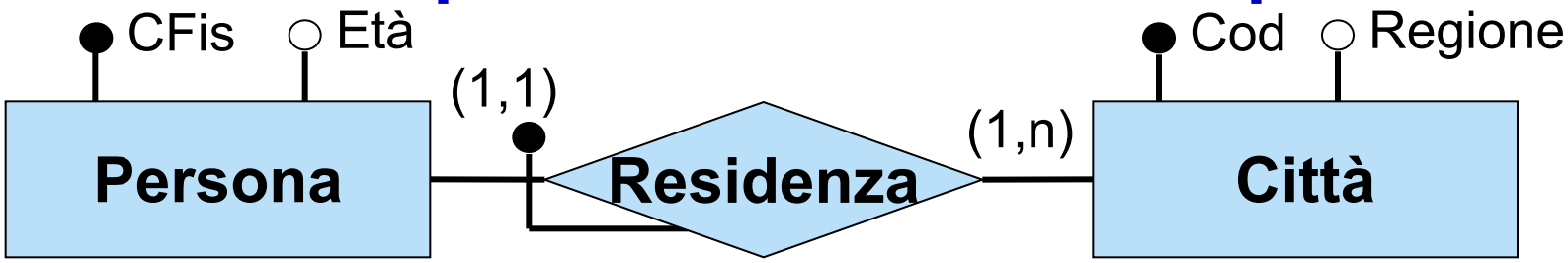
Si noti che **K1** e **K2** sono chiavi, anche non primarie. Le due tabelle sono fortemente accoppiate, nel senso che valgono le foreign key reciproche.

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono **R1** o **R2** vanno riformulati su **R**

- Si applica per facilitare gli accessi a **R1** e **R2** quando questi avvengono prevalentemente insieme e richiedono di calcolare il join tra **R1** e **R2** con **K1=K2**; in altre parole si applica per evitare il join tra **R1** e **R2**
- Le relazioni **R1** e **R2** possono essere ricostruite attraverso due viste che calcolano rispettivamente le proiezioni di **R** su **(K, α)** e su **(K, β)**
- Quando non c'è β, serve ad eliminare una relazione inutile (cioè **R2**), che può essere ricostruita attraverso la proiezione di **R** su **K** (vedi dopo)

Accorpamento caso 1 - esempio 1



Persona(CFis, Età)

foreign key: Persona[CFis] $\subseteq$ Residenza[Persona]

Residenza(Persona, Città)

foreign key: Residenza[Persona] $\subseteq$ Persona[CFis]

foreign key: Residenza[Città] $\subseteq$ Città[Cod]

Città(Cod, Regione)

inclusione: Città[Cod] $\subseteq$ Residenza[Città]

Supponiamo che quando si accede ad una persona si acceda spesso anche alla sua città di residenza. Applichiamo quindi l'accorpamento, ed otteniamo:

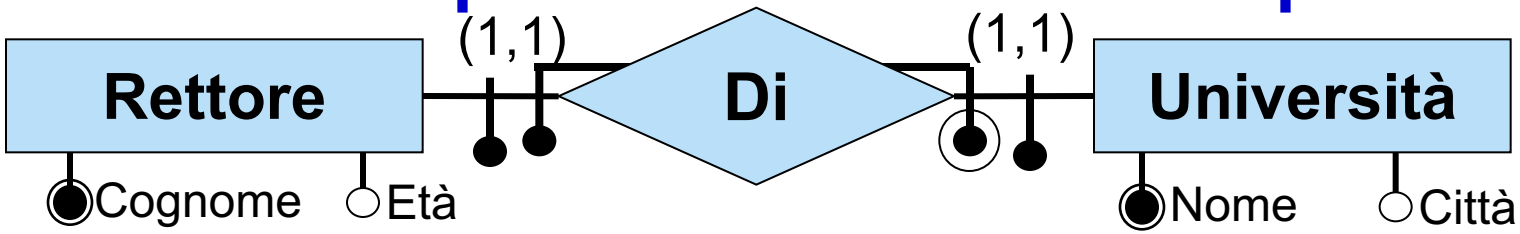
Persona(CFis, Età, Città)

foreign key: Persona[Città] $\subseteq$ Città[Cod]

Città(Cod, Regione)

inclusione: Città[Cod] $\subseteq$ Persona[Città]

Accorpamento caso 1 - esempio 2



Rettore(Cognome, Età)
  foreign key: Rettore[Cognome] ⊆ Di[Rettore]
Di(Rettore, Università)
  foreign key: Di[Rettore] ⊆ Rettore[Cognome]
  foreign key: Di[Università] ⊆ Università[Nome]
  chiave: Rettore
Università(Nome, Città)
  foreign key: Università[Nome] ⊆ Di[Università]

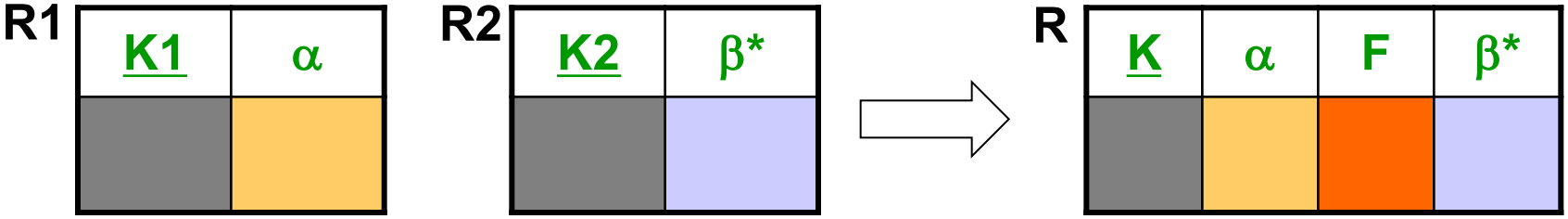
Supponiamo che quando si accede ad una università si acceda anche sempre al suo rettore. Applichiamo l'accorpamento di Università e Di, ed otteniamo

Rettore(Cognome, Età)
  foreign key: Rettore[Cognome] ⊆ Università[Rettore]
Università(Nome, Città, Rettore)
  foreign key: Università[Rettore] ⊆ Rettore[Cognome]
  chiave: Rettore

Supponiamo che quando si accede ad un rettore si acceda sempre anche ai dati relativi alla sua università. Applichiamo un ulteriore accorpamento ed otteniamo

UniversitàRettore(NomeUniversità, CittàUniversità, CognomeRettore, EtàRettore)
  chiave: CognomeRettore

Accorpamento per facilitare l'accesso: caso 2



foreign key: R2[K2] ⊆ R1[K1]

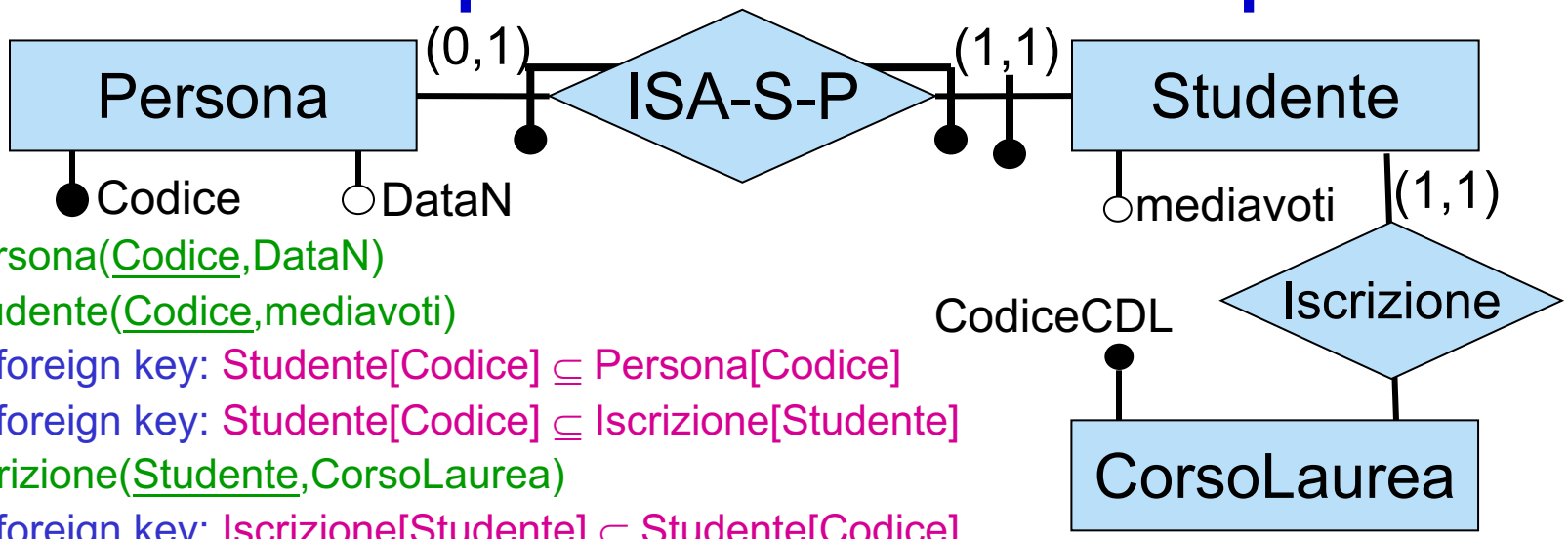
Si noti che **K1** e **K2** sono chiavi, anche non primarie. Le due tabelle sono debolmente accoppiate, nel senso che vale una sola foreign key (in questo caso da **R2** a **R1**). L'accorpamento prevede di accorpare **R2** a **R1** ottenendo **R**: l'attributo **F** (flag) è booleano, e vale true nelle tuple di **R** corrispondenti a tuple di **R2**, false nelle altre. Se β è not null in **R2**, allora **F** è ridondante (perché **F** sarebbe true se e solo se β non è null in **R1**).

Vincoli dello schema ristrutturato:

- vincolo che dice che se **F** è false, allora β è NULL
- tutti i vincoli che coinvolgono **R1** o **R2** vanno riformulati su **R**

- Si applica per facilitare gli accessi a **R1** e **R2** quando questi avvengono prevalentemente insieme e richiedono di calcolare il join tra **R1** e **R2** con **K1=K2**; in altre parole si applica per evitare il join tra **R1** e **R2**
- Le relazioni **R1** e **R2** possono essere ricostruite attraverso due viste che calcolano rispettivamente la proiezioni di **R** su **(K, α)** e la proiezione su **(K, β)** solo per quelle tuple per cui **F** è true.
- Si può eseguire anche quando β manca: in questo caso serve ad eliminare una relazione (**R2**), che peraltro può essere ricostruita mediante l'attributo **F**.

Accorpamento caso 2: esempio 1



Persona(Codice,DataN)

Studente(Codice,mediavoti)

foreign key: Studente[Codice] $\subseteq$ Persona[Codice]

foreign key: Studente[Codice] $\subseteq$ Iscrizione[Studente]

Iscrizione(Studente,CorsoLaurea)

foreign key: Iscrizione[Studente] $\subseteq$ Studente[Codice]

foreign key: Iscrizione[CorsoLaurea] $\subseteq$ CorsoLaurea[CodiceCDL]

CorsoLaurea(CodiceCDL)

Supponiamo che le indicazioni di progetto ci dicano che ogni volta che si accede ad uno studente si vuole sempre sapere la sua data di nascita e ogni volta che si accede ad una persona si vuole sapere la sua media dei voti, se è studente. A questo punto possiamo optare per l'accorpamento di Studente (debolmente accoppiata a Persona) a Persona. Otteniamo così il seguente schema ristrutturato.

Persona(Codice,DataN,mediavoti*)

inclusione $SEL_{\text{mediavoti is not null}}$ Persona[Codice] $\subseteq$ Iscrizione[Studente]

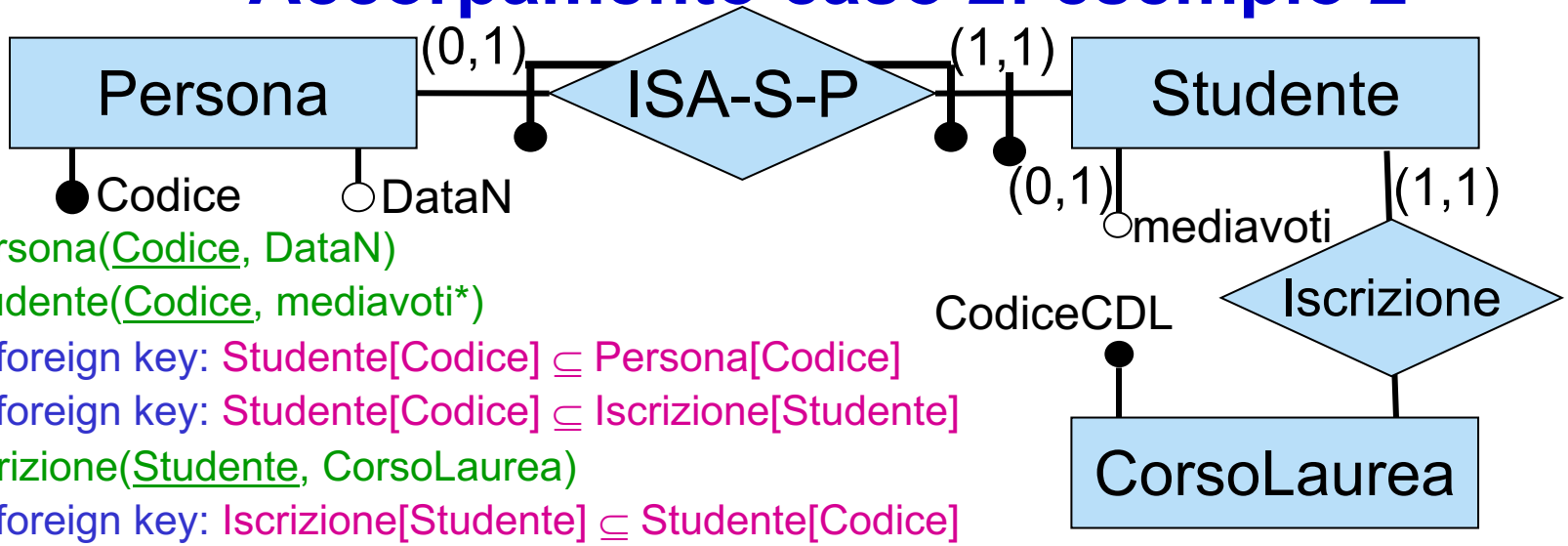
Iscrizione(Studente,CorsoLaurea)

inclusione: Iscrizione[Studente] $\subseteq SEL_{\text{mediavoti is not null}}$ Persona[Codice]

CorsoLaurea(CodiceCDL)

L'attributo booleano flag non è necessario, visto che per discriminare le tuple relative agli studenti basta la condizione "mediavoti is not null"

Accorpamento caso 2: esempio 2



Persona(Codice, DataN)

Studente(Codice, mediavoti*)

foreign key: Studente[Codice] $\subseteq$ Persona[Codice]

foreign key: Studente[Codice] $\subseteq$ Iscrizione[Studente]

Iscrizione(Studente, CorsoLaurea)

foreign key: Iscrizione[Studente] $\subseteq$ Studente[Codice]

foreign key: Iscrizione[CorsoLaurea] $\subseteq$ CorsoLaurea[CodiceCDL]

CosoLaurea(CodiceCDL)

Supponiamo che le indicazioni di progetto ci dicano che ogni volta che si accede ad uno studente si vuole sempre sapere la sua data di nascita e ogni volta che si accede ad una persona si vuole sapere la sua media dei voti, se è studente. A questo punto possiamo optare per l'accorpamento di Studente (debolmente accoppiata a Persona) a Persona. Considerando che mediavoti questa volta è opzionale in Studente, otteniamo:

Persona(Codice, DataN, flag, mediavoti*)

vincolo di tupla: se flag=false allora mediavoti IS NULL

inclusione: $SEL_{flag=true}$ Persona[Codice] $\subseteq$ Iscrizione[Studente]

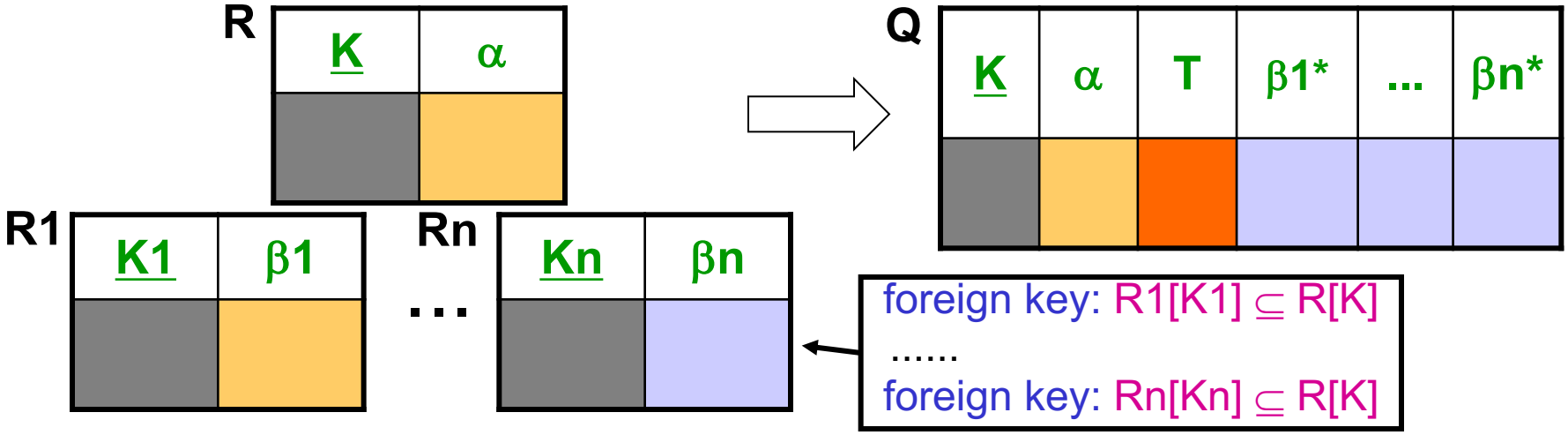
Iscrizione(Studente, CorsoLaurea)

inclusione: Iscrizione[Studente] $\subseteq$ $SEL_{flag=true}$ Persona[Codice]

CorsoLaurea(CodiceCDL)

L'attributo booleano flag serve per discriminare le tuple relative agli studenti (flag = true). Infatti in questo caso la condizione "mediavoti is not null" non basta, visto che mediavoti può essere null anche per tali tuple

Accorpamento per facilitare l'accesso: caso 3



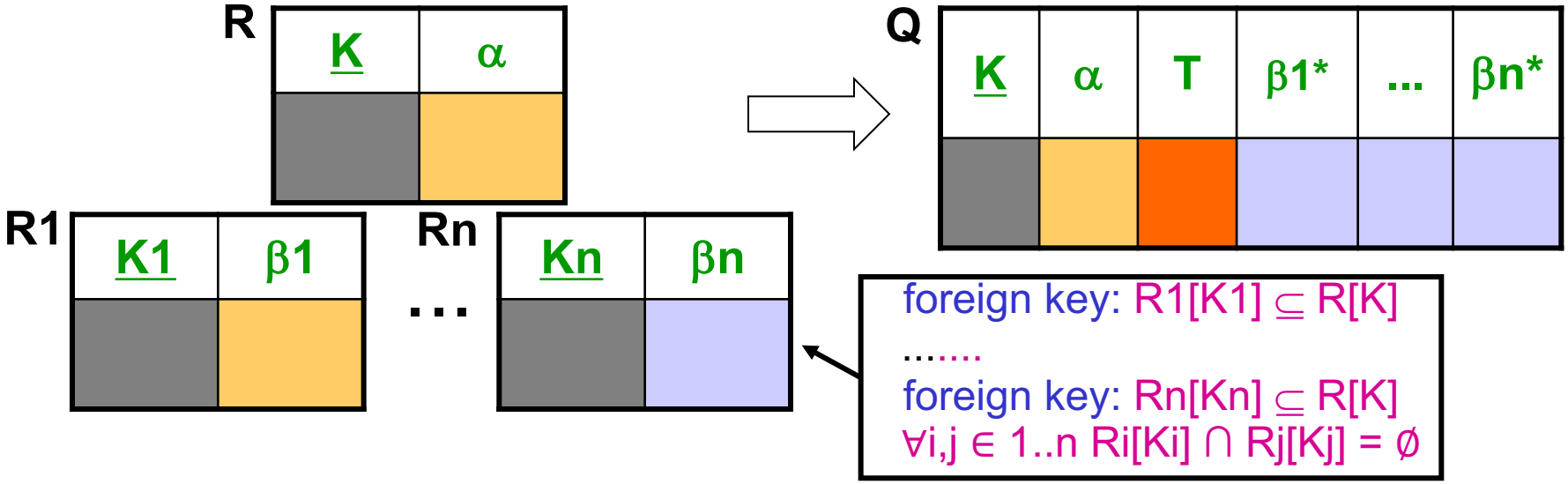
Questo caso può essere visto come "accorpamento orizzontale", l'inverso della decomposizione orizzontale. **T** (che sta per Tipo) rappresenta o un vettore di n attributi booleani oppure un attributo **T** enumerato che può assumere 2^n valori, ciascuna corrispondente ad una combinazione true o false per **R1**,..., **Rn**. Ossia, esso vale "i" nelle tuple corrispondenti a tuple che stanno nella combinazione (ad esempio in R1, R2 e non in R3 e R4) dei vari **Ri** denotata da i. Ogni tabella **Ri** è debolmente accoppiata, con **R**, nel senso che, per ogni i, vale la foreign key da **Ri** a **R** (e non quella da **R** a **Ri**). Ne segue che tutte le tabelle **Ri** si accorpano a **R**.

Vincoli dello schema ristrutturato:

- vincoli che mettono in relazione il valore di **T** con le tuple di **R1**,..., **Rn** e quindi anche i possibili valori nulli in **β1 … βn** (ad esempio, se **β1** non è NULL, allora **T** ha un certo valore)
- tutti i vincoli che coinvolgono **R1** o **R2** vanno riformulati su **R**

- Si applica per facilitare gli accessi a **Ri** e **R** quando questi avvengono prevalentemente insieme. Senza l'accorpamento tali accessi richiederebbero di calcolare il join tra **R** e **Ri** con **K=Ki**; in altre parole si applica per evitare il join tra **R** e le varie **Ri**.
- Le relazioni **Ri** possono essere ricostruite attraverso viste.

Accorpamento per facilitare l'accesso: caso 3a

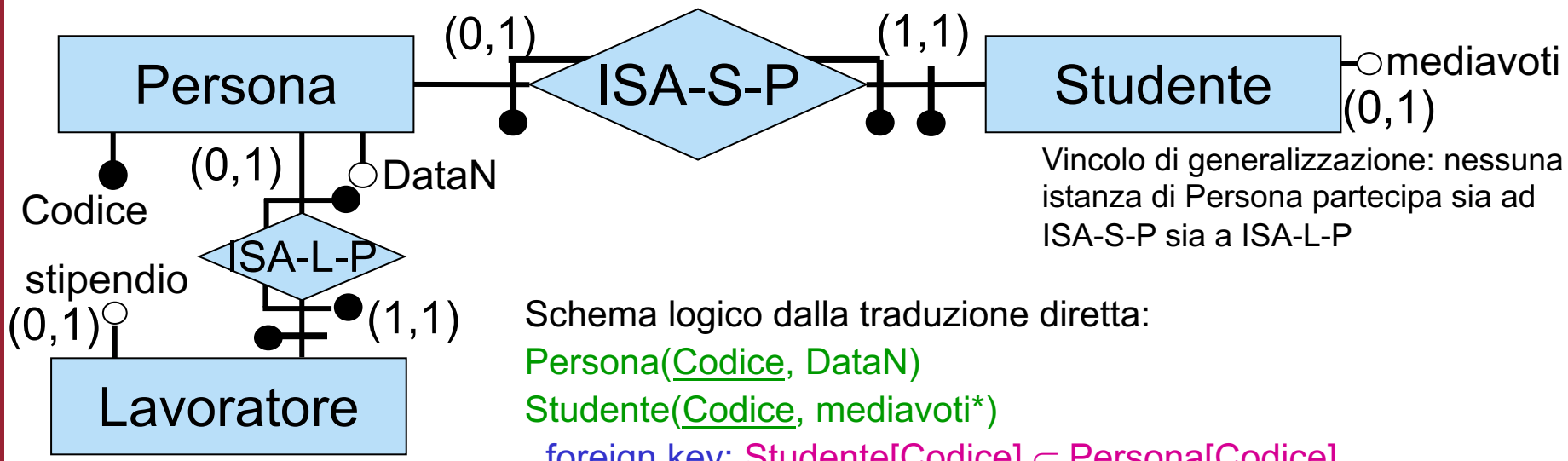


Questo caso è un caso particolare del caso 3, ossia il caso in cui le varie **Ri[Ki]** sono disgiunte a coppie. L'attributo **T** (tipo) questa volta può assumere i valori 1...n corrispondenti a **R1**,..., **Rn** (ovvero, questa volta bastano log_n bits): esso vale "i" nelle tuple corrispondenti a tuple di **Ri**. Come prima, ogni tabella **Ri** è debolmente accoppiata con **R**, ossia, vale la foreign key da **Ri** a **R** (e non quella da **R** a **Ri**). E come prima, tutte le tabelle **Ri** si accorpano a **R**.

Vincoli dello schema ristrutturato:

- vincoli che mettono in relazione il valore di **T** con le tuple di **R1**,..., **Rn** e quindi anche i possibili valori nulli in **β1 … βn** (ad esempio, se **β1** non è NULL, allora **T** ha valore 1)
- tutti i vincoli che coinvolgono **R1** o **R2** vanno riformulati su **R**

Accorpamento caso 3a: esempio



Vincolo di generalizzazione: nessuna istanza di Persona partecipa sia ad ISA-S-P sia a ISA-L-P

Schema logico dalla traduzione diretta:

Persona(Codice, DataN)

Studente(Codice, mediavoti*)

foreign key: Studente[Codice] $\subseteq$ Persona[Codice]

Lavoratore(Codice, stipendio*)

foreign key: Lavoratore[Studente] $\subseteq$ Persona[Codice]

vincolo esterno: Lavoratore[Codice] $\cap$ Studente[Codice] = $\emptyset$

Supponiamo che le indicazioni di progetto ci dicano che ogni volta che si accede ad uno studente o ad un lavoratore si vuole sempre sapere la sua data di nascita ed ogni volta che si accede ad una persona si vuole sapere l'eventuale media dei voti (se studente) o l'eventuale stipendio (se lavoratore). A questo punto possiamo optare per l'accorpamento a Persona delle due tabelle Studente e Lavoratore (entrambe accoppiate debolmente a Persona):

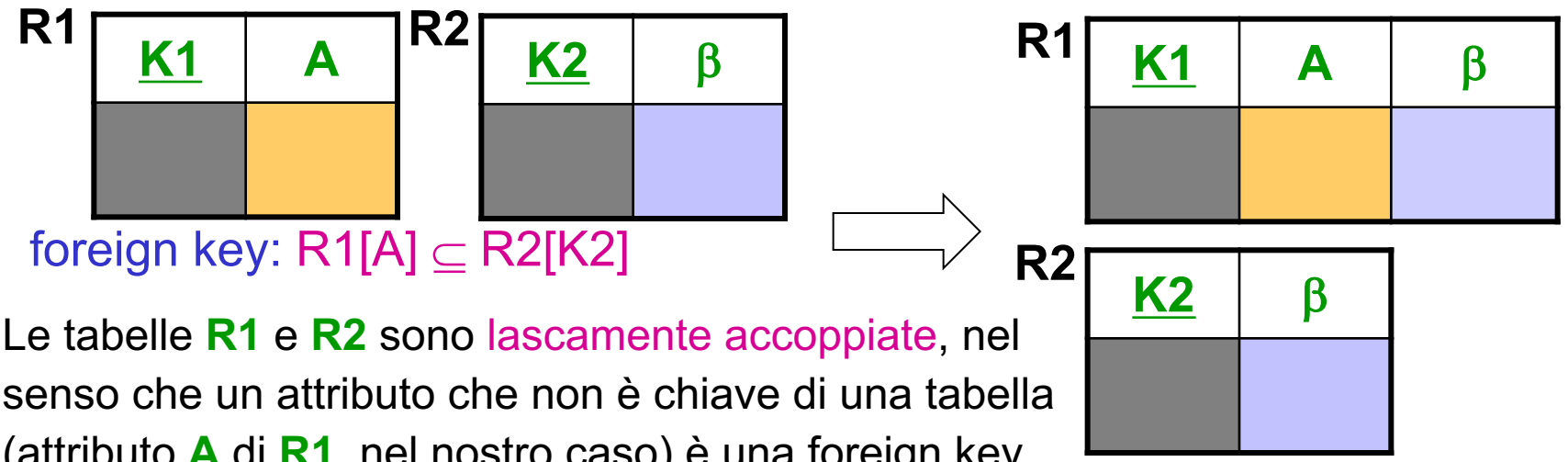
Persona(Codice, DataN, Tipo, mediavoti*, stipendio*)

vincolo di dominio: Tipo assume valori "studente" o "lavoratore"

vincolo di tupla: se mediavoti is not null allora Tipo = "studente"

vincolo di tupla: se stipendio is not null allora Tipo = "lavoratore"

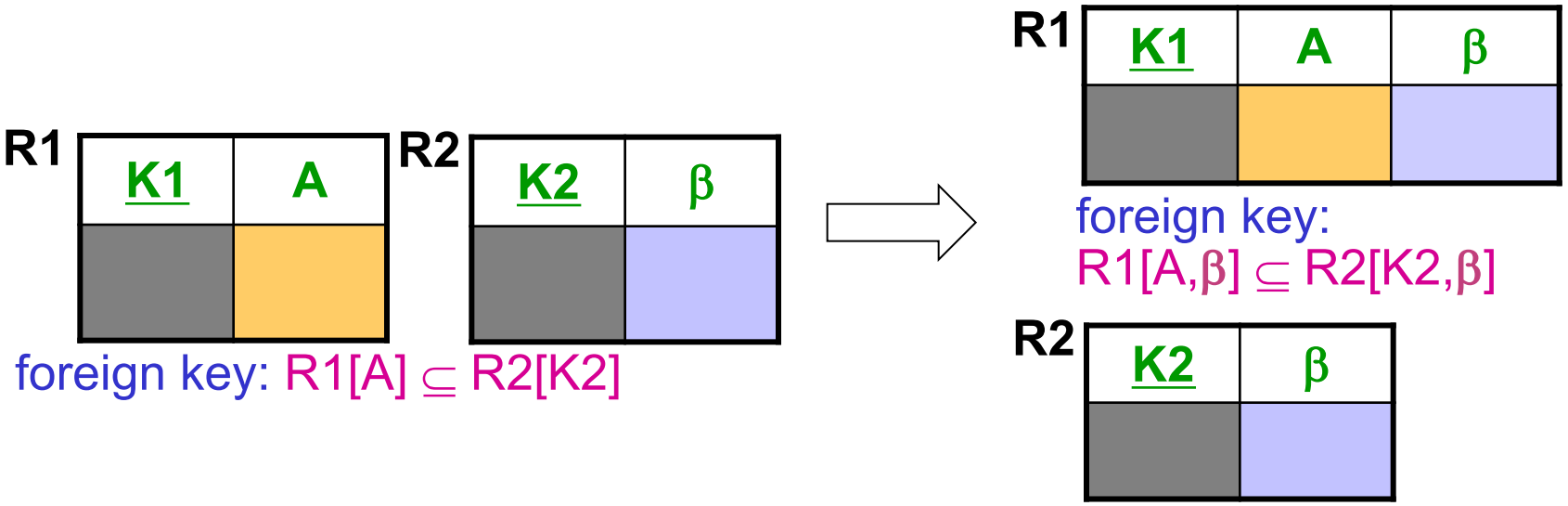
Accorpamento per facilitare l'accesso: caso 4



Le tabelle **R1** e **R2** sono lascamente accoppiate, nel senso che un attributo che non è chiave di una tabella (attributo **A** di **R1**, nel nostro caso) è una foreign key dell'altra (**R2**, nel nostro caso) o, in alcuni casi, anche quando è solo incluso in un attributo dell'altra.
L'operazione prevede di accorpare gli attributi β di **R2** ad **R1**, ma senza eliminare la tabella **R2**.

Questo tipo di accorpamento si chiama "accorpamento per denormalizzazione" e si fa in condizioni molto particolari, perché crea ridondanze che possono dare problemi. In pratica si "denormalizza" (così si dice, si veda dopo) la relazione **R1** inserendo in essa un insieme di attributi β che dipendono non dalla chiave di **R1** ma da un suo campo (o da suoi campi) non chiave (in questo caso, **A**). Si applica quando l'accesso a **R1** richiede sempre (o spesso) il join con **R2** sulla condizione **A** = **K2**.

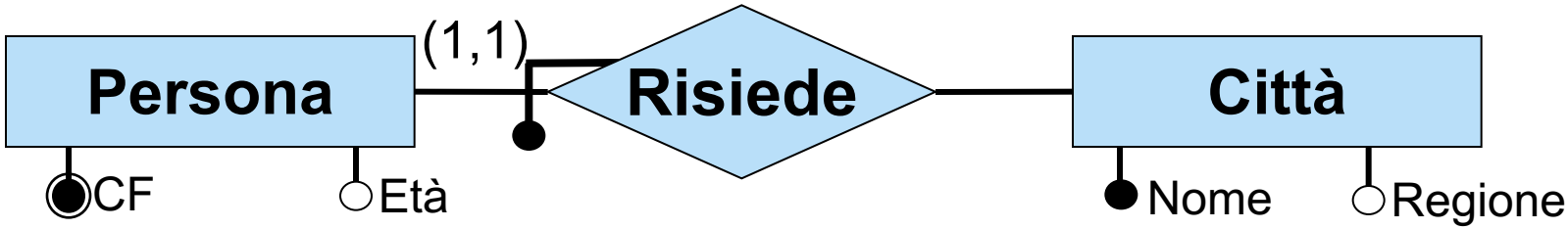
Accorpamento per facilitare l'accesso: caso 4



Tipicamente, l'accorpamento per denormalizzazione non è problematico quando i valori di β nella tabella **R2** non vengono modificati o vengono modificati raramente. In queste condizioni si ha solo il vantaggio che la nuova tabella **R1**, inglobando l'attributo o gli attributi β, evita il join. Si noti che la tabella **R1** originaria può essere ricostruita attraverso la proiezione della nuova tabella **R1** sugli attributi **K1,A.**

Sotto particolari condizioni nello schema originario (ad esempio, con una inclusione da **R2[K2]** a **R1[A]** nello schema originario che ovviamente si riporta anche nello schema di arrivo), si può anche eliminare nello schema di arrivo la relazione **R2**, perché diventa inutile (si veda dopo). In questo caso, però, occorrerebbe aggiungere un vincolo esterno su **R1** che asserisce che β dipende funzionalmente da **A**.

Accorpamento caso 4 - esempio 1



Persona(CF, Età)
  foreign key: Persona[CF] ⊆ Risiede[Persona]
Risiede(Persona, Città)
  foreign key: Risiede[Persona] ⊆ Persona[CF]
  foreign key: Risiede[Città] ⊆ Città[Nome]
Città(Nome, Regione)

Supponiamo che (1) quando si accede ad una persona si vuole sempre conoscere la città di residenza e (2) quando si accede ad una città di residenza di una persona si vuole sempre conoscere la regione di tale città.

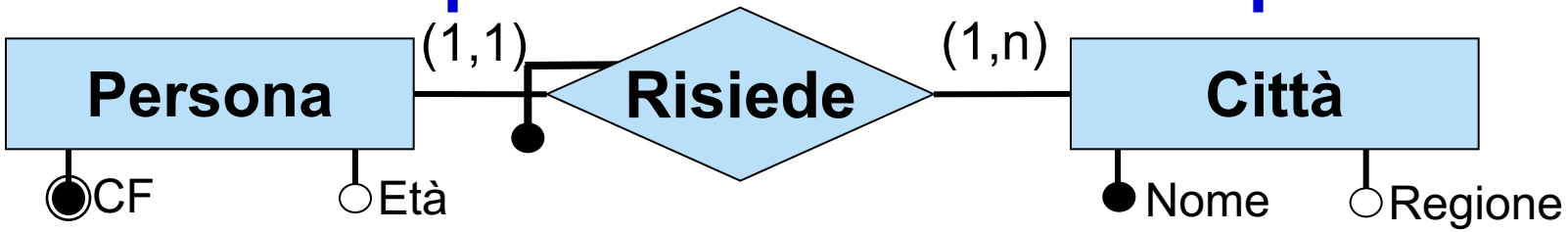
Per (1), applichiamo l'accorpamento di Persona e Risiede, ottenendo

Persona(CF, Età, Città)
  foreign key: Persona[Città] ⊆ Città[Nome]
Città(Nome, Regione)

Per 2, applichiamo l'accorpamento per denormalizzazione, ottenendo

Persona(CF, Età, Città, Regione)
  inclusione: Persona[Città, Regione] ⊆ Città[Nome,Regione]
Città(Nome, Regione)

Accorpamento caso 4 - esempio 2



Persona(CF, Età)
foreign key: Persona[CF] ⊆ Risiede[Persona]
foreign key: Risiede[Persona] ⊆ Persona[CF]
Città(Nome, Regione)
inclusione: Città[Nome] ⊆ Risiede[Città]

Risiede(Persona, Città)
foreign key: Risiede[Città] ⊆ Città[Nome]

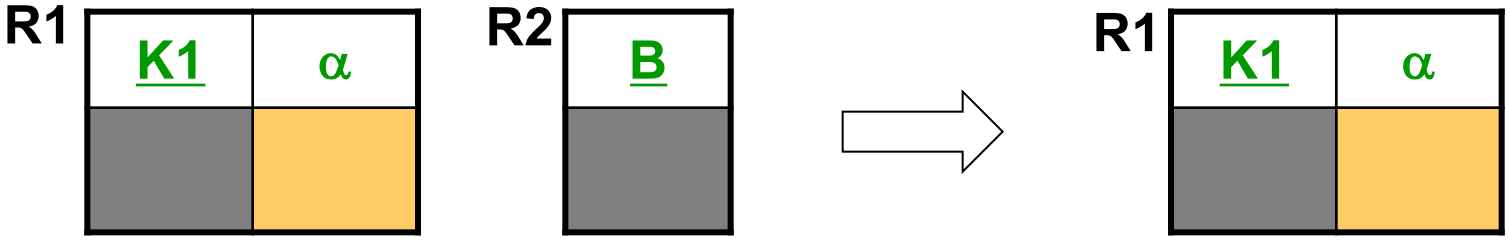
Supponiamo che (1) quando si accede ad una persona si vuole sempre conoscere la città di residenza e (2) quando si accede ad una persona si vuole sempre conoscere la regione della città della persona. Come visto prima, ecco cosa otteniamo applicando due accorpamenti:

Persona(CF, Età, Città, Regione)
inclusione: Persona[Città, Regione] ⊆ Città[Nome,Regione]
Città(Nome, Regione)
inclusione: Città[Nome, Regione] ⊆ Persona[Città, Regione]

In questo caso, però, la cardinalità minima su Città induce anche l'inclusione tra la tabella Città e la tabella Persona. A questo punto si può applicare l'accorpamento discusso nel caso 4, eliminando la tabella Città e ottenendo

Persona(CF, Età, Città, Regione)
vincolo esterno: non esistono due tuple in Persona con la stessa città e diversa regione

Accorpamento per eliminare relazioni inutili



foreign key: $R1[\alpha] \subseteq R2[B]$

inclusione: $R2[B] \subseteq R1[\alpha]$

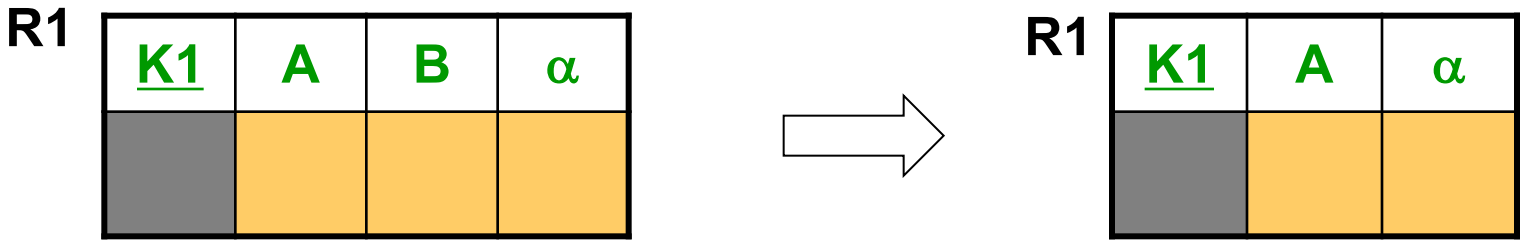
Si noti che **B** è chiave in **R2**, ed α può esserlo (o può contenere una chiave) o no in **R1** (nella figura è mostrato il caso in cui non lo sia, ma la regola vale anche nel caso in cui α sia chiave o parte di chiave – se è chiave, il caso corrisponde all'accorpamento - caso 2).

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono **R2** vanno riformulati su **R1**

- Si applica quando è raro che interessi solo **B** avulso da **K1** e serve ad eliminare la relazione **R2** che è a questo punto è inutile, visto che tutti i valori negli attributi α (e quindi anche **B**) si ritrovano in α di **R1**
- La relazione **R2** può essere ricostruita attraverso la proiezione della relazione **R1** sugli attributi α

Accorpamento per eliminare attributi inutili



Vincolo di tupla: per ogni tupla t di R1, t[A]=t[B]

Si noti che **A,B** sono attributi qualunque, oppure insiemi di attributi (in questo caso, t[A] e t[B] saranno tuple, e non singoli valori). Uno dei due attributi, ad esempio B è chiaramente inutile.

Vincoli dello schema ristrutturato:

- tutti i vincoli che coinvolgono **B** vanno riformulati su **A**

- È un accorpamento di attributi, piuttosto che di tabelle. Si applica per eliminare attributi inutili
- La relazione originale può essere ricostruita attraverso un'opportuna vista, che per ogni tupla della relazione "duplica" il valore di **A** in un nuovo attributo.

Un altro esempio di eliminazione di attributo inutile è quello in cui un attributo della relazione è sempre NULL: in questo caso si elimina semplicemente l'attributo.

Traduzione in SQL

Ogni relazione viene definita tramite la “create table”, nel modo ovvio. In particolare, i vincoli vengono tradotti secondo queste indicazioni generali:

- attributi obbligatori: `not null`
- chiave primaria: `primary key`
- chiave: `unique`
- foreign key: `foreign key`
- interdipendenza valori nulli:

```
check ((A is null and B is null) or
       (A is not null and B is not null))
```

- inclusione: `check (A in (select B from R))`
- disgiuntezza: `check (A not in (select B from R))`
- cardinalità (i,j):

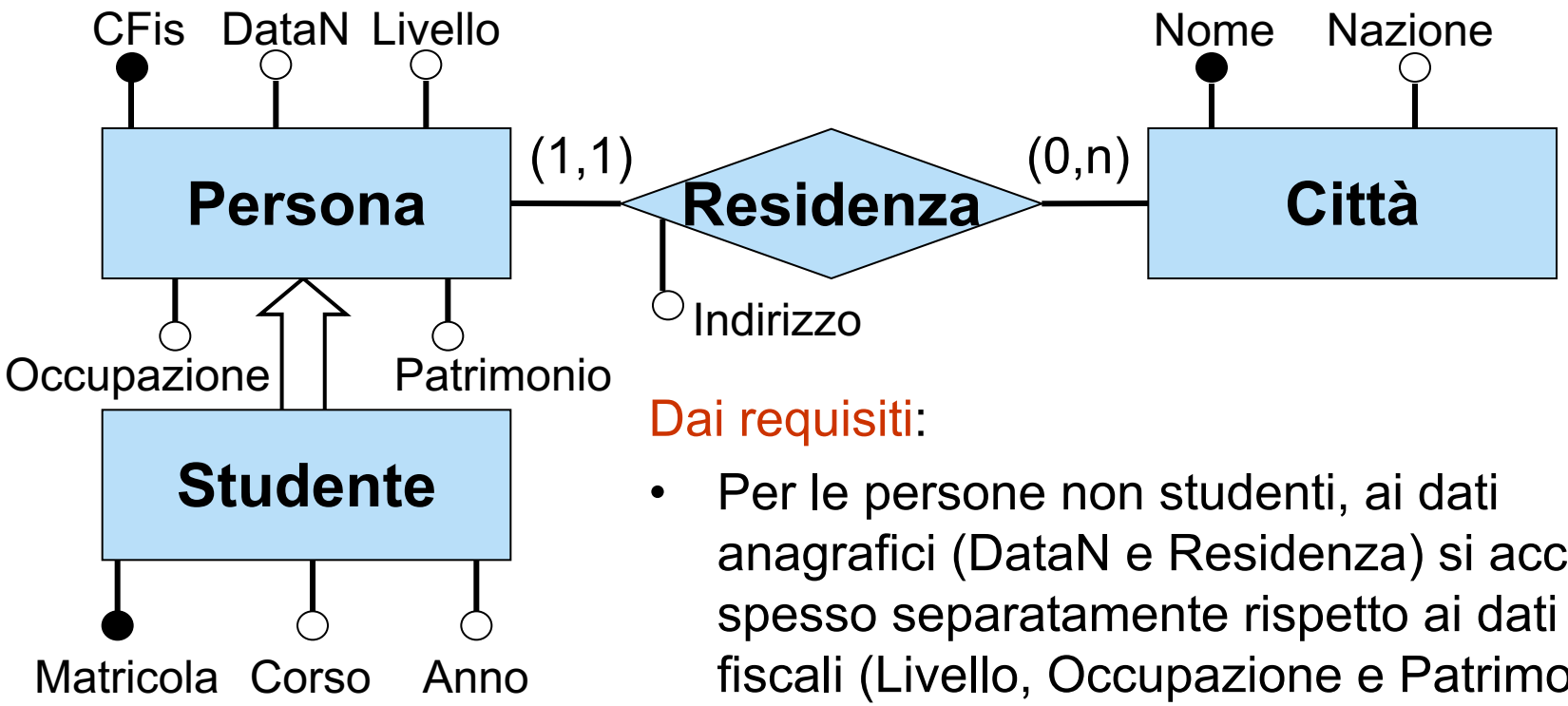
```
check ((i ≤ (select count(*) from R where ...)) and
       (j ≥ (select count(*) from R where ...)))
```

- vincoli esterni: `assertion` o controllati a livello di applicazione

Se il DBMS non consente check intra-relazionali (come purtroppo tutti quelli attuali), allora si possono usare i trigger per rappresentarli nello schema.

Esercizio 7: effettuare la progettazione logica

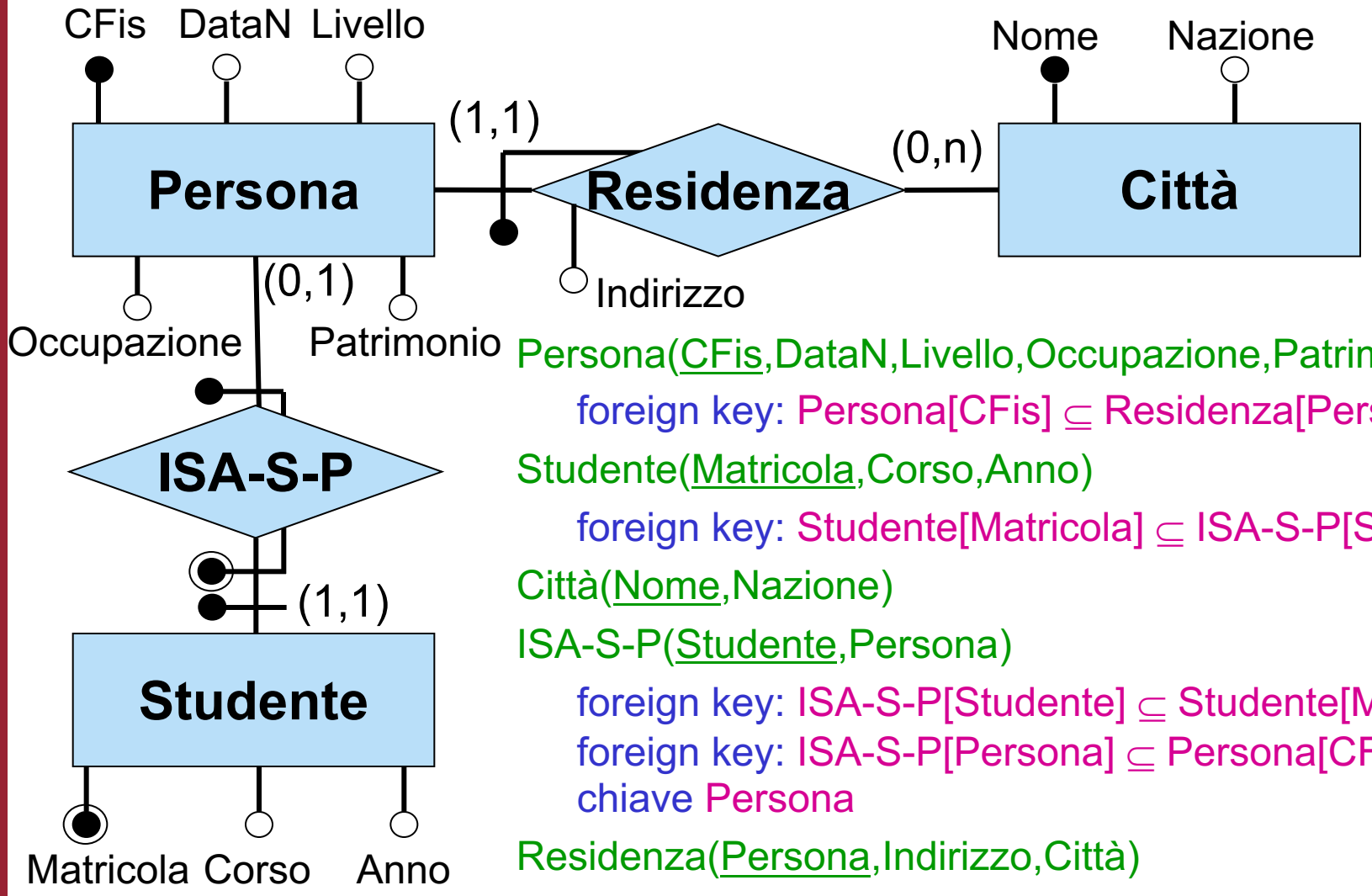
Schema concettuale:



Dai requisiti:

- Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio)
- Agli studenti si accede prevalentemente per matricola, e si accede spesso ai dati universitari (Corso, Anno) insieme alla data di nascita e ai dati fiscali (Livello, Occupazione e Patrimonio)

Esercizio 7: ristrutturazione e traduzione diretta



Persona(CFis,DataN,Livello,Occupazione,Patrimonio)

foreign key: Persona[CFis] $\subseteq$ Residenza[Persona]

Studente(Matricola,Corso,Anno)

foreign key: Studente[Matricola] $\subseteq$ ISA-S-P[Studente]

Città(Nome,Nazione)

ISA-S-P(Studente,Persona)

foreign key: ISA-S-P[Studente] $\subseteq$ Studente[Matricola]

foreign key: ISA-S-P[Persona] $\subseteq$ Persona[CFis]

chiave Persona

Residenza(Persona,Indirizzo,Città)

foreign key: Residenza[Persona] $\subseteq$ Persona[CFis]

foreign key: Residenza[Città] $\subseteq$ Città[Nome]

Esercizio 7: ristrutturazioni dello schema logico

Per le persone non studenti valgono criteri di accesso diversi rispetto alle persone che sono studenti, anche relativamente ai dati riguardanti la residenza:

- **decomposizione orizzontale** di Persona in PersonaNonStudente e PersonaStudente
- **decomposizione orizzontale** di Residenza in ResidenzaNonStudente e ResidenzaStudente

PersonaNonStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
foreign key: PersonaNonStudente[CFis] $\subseteq$ ResidenzaNonStudente[Persona]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
foreign key: PersonaStudente[CFis] $\subseteq$ ISA-S-P[Persona]
foreign key: PersonaStudente[CFis] $\subseteq$ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
foreign key: Studente[Matricola] $\subseteq$ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
foreign key: ISA-S-P[Studente] $\subseteq$ Studente[Matricola]
foreign key: ISA-S-P[Persona] $\subseteq$ PersonaStudente[CFis]
chiave: Persona
ResidenzaNonStudente(Persona,Indirizzo,Città)
foreign key: ResidenzaNonStudente[Persona] $\subseteq$ PersonaNonStudente[CFis]
foreign key: ResidenzaNonStudente[Città] $\subseteq$ Città[Nome]
ResidenzaStudente(Studente,Indirizzo,Città)
foreign key: ResidenzaStudente[Studente] $\subseteq$ PersonaStudente[CFis]
foreign key: ResidenzaStudente[Città] $\subseteq$ Città[Nome]
Vincolo: PersonaNonStudente[CFis] $\cap$ PersonaStudente[CFis] = $\varnothing$

Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio):

- **decomposizione verticale** di PersonaNonStudente in PersonaDatiAnagrafe e PersonaDatiFisco

PersonaDatiAnagrafe(CFis,DataN)
  foreign key: PersonaDatiAnagrafe[CFis] $\subseteq$ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[CFis] $\subseteq$ ResidenzaNonStudente[Persona]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] $\subseteq$ PersonaDatiAnagrafe[CFis]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaStudente[CFis] $\subseteq$ ISA-S-P[Persona]
  foreign key: PersonaStudente[CFis] $\subseteq$ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] $\subseteq$ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] $\subseteq$ Studente[Matricola]
  foreign key: ISA-S-P[Persona] $\subseteq$ PersonaStudente[CFis] chiave: Persona
ResidenzaNonStudente(Persona,Indirizzo,Città)
  foreign key: Residenza[Persona] $\subseteq$ PersonaDatiAnagrafe[CFis]
  foreign key: Residenza[Città] $\subseteq$ Città[Nome]
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] $\subseteq$ PersonaStudente[CFis]
  foreign key: Residenza[Città] $\subseteq$ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] $\cap$ PersonaStudente[CFis] = $\varnothing$

Per le persone non studenti, ai dati anagrafici (DataN e Residenza) si accede spesso separatamente rispetto ai dati fiscali (Livello, Occupazione e Patrimonio):

- **accorpamento** tra PersonaDatiAnagrafe e ResidenzaNonStudente

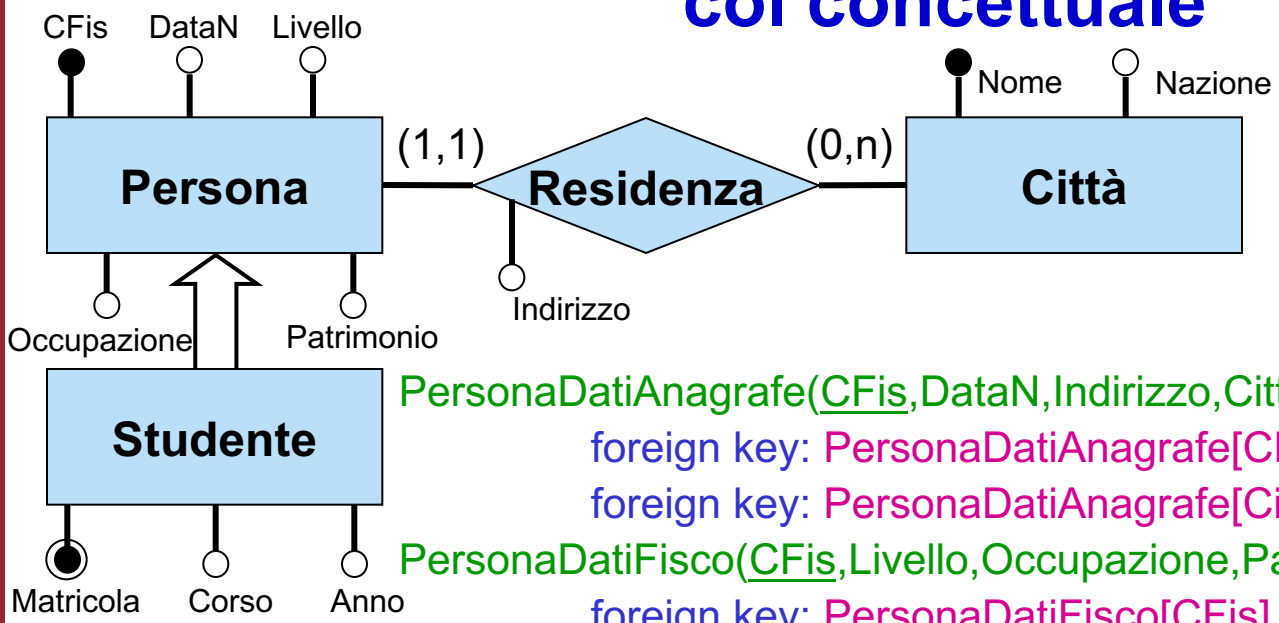
PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)
  foreign key: PersonaDatiAnagrafe[CFis] $\subseteq$ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[Città] $\subseteq$ Città[Nome]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] $\subseteq$ PersonaDatiAnagrafe[CFis]
PersonaStudente(CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: PersonaStudente[CFis] $\subseteq$ ISA-S-P[Persona]
  foreign key: PersonaStudente[CFis] $\subseteq$ ResidenzaStudente[Studente]
Studente(Matricola,Corso,Anno)
  foreign key: Studente[Matricola] $\subseteq$ ISA-S-P[Studente]
Città(Nome,Nazione)
ISA-S-P(Studente,Persona)
  foreign key: ISA-S-P[Studente] $\subseteq$ Studente[Matricola]
  foreign key: ISA-S-P[Persona] $\subseteq$ PersonaStudente[CFis] chiave: Persona
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] $\subseteq$ PersonaStudente[CFis]
  foreign key: Residenza[Città] $\subseteq$ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] $\cap$ PersonaStudente[CFis] = $\varnothing$

Agli studenti si accede prevalentemente per matricola, e si accede spesso ai dati universitari (Corso, Anno) insieme alla data di nascita e ai dati fiscali (Livello, Occupazione e Patrimonio):

- **accorpamento** tra Studente e ISA-S-P, e poi tra la relazione così ottenuta e PersonaStudente

PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)
  foreign key: PersonaDatiAnagrafe[CFis] $\subseteq$ PersonaDatiFisco[CFis]
  foreign key: PersonaDatiAnagrafe[Città] $\subseteq$ Città[Nome]
PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)
  foreign key: PersonaDatiFisco[CFis] $\subseteq$ PersonaDatiAnagrafe[CFis]
Studente(Matricola,Corso,Anno,CFis,DataN,Livello,Occupazione,Patrimonio)
  foreign key: Studente[CFis] $\subseteq$ ResidenzaStudente[Studente]
  chiave: CFis
Città(Nome,Nazione)
ResidenzaStudente(Studente,Indirizzo,Città)
  foreign key: ResidenzaStudente[Studente] $\subseteq$ Studente[CFis]
  foreign key: Residenza[Città] $\subseteq$ Città[Nome]
Vincolo: PersonaDatiAnagrafe[CFis] $\cap$ Studente[CFis] = $\varnothing$

Esercizio 7: schema logico e confronto col concettuale



PersonaDatiAnagrafe(CFis,DataN,Indirizzo,Città)

foreign key: PersonaDatiAnagrafe[CFis] $\subseteq$ PersonaDatiFisco[CFis]

foreign key: PersonaDatiAnagrafe[Città] $\subseteq$ Città[Nome]

PersonaDatiFisco(CFis,Livello,Occupazione,Patrimonio)

foreign key: PersonaDatiFisco[CFis] $\subseteq$ PersonaDatiAnagrafe[CFis]

Studente(Matricola,Corso,Anno,CFis,DataN,Livello,Occupazione,Patrimonio)

foreign key: Studente[CFis] $\subseteq$ ResidenzaStudente[Studente]

chiave: CFis

ResidenzaStudente(Studente,Indirizzo,Città)

foreign key: ResidenzaStudente[Studente] $\subseteq$ Studente[CFis]

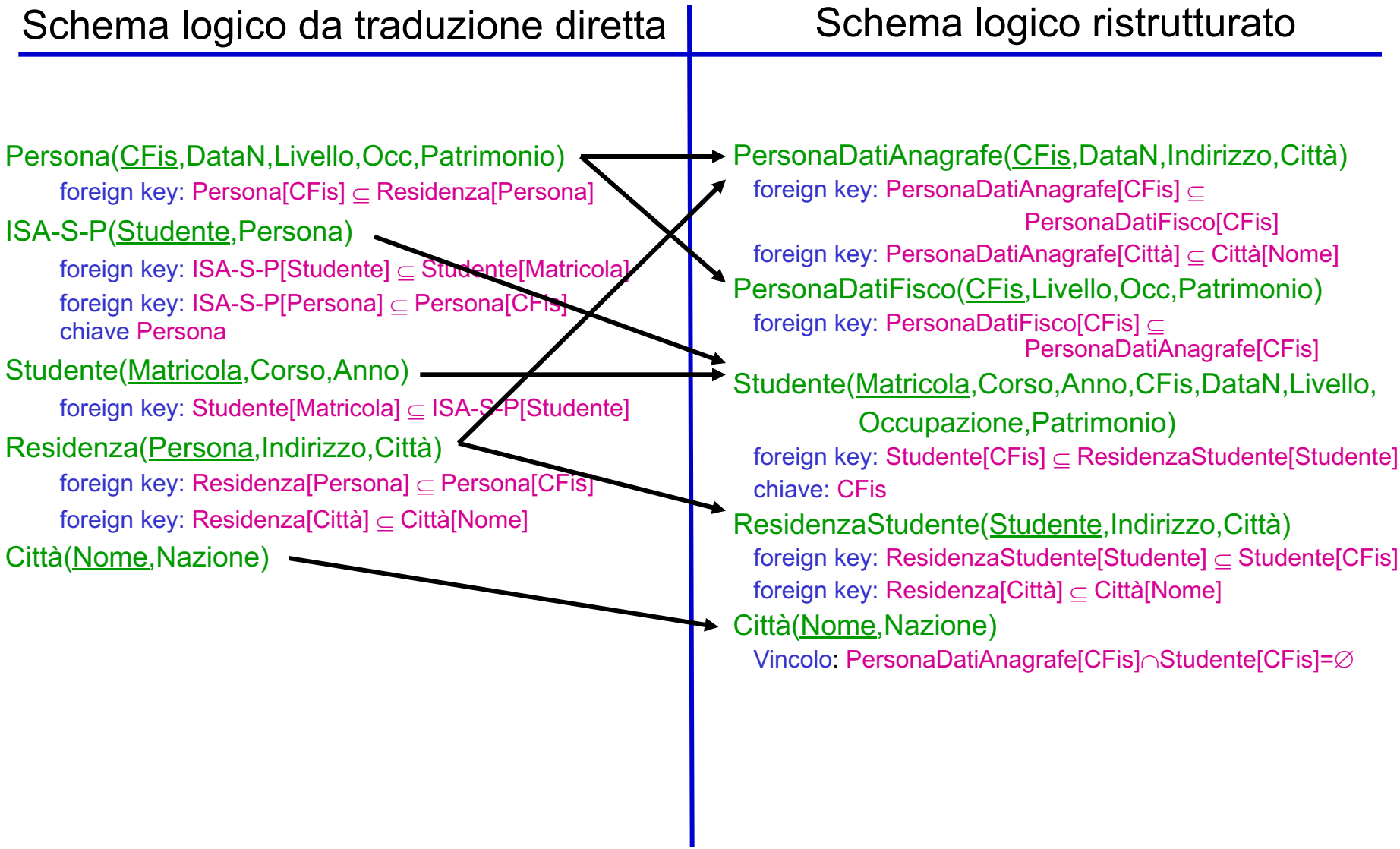
foreign key: Residenza[Città] $\subseteq$ Città[Nome]

Città(Nome,Nazione)

Vincolo: PersonaDatiAnagrafe[CFis] $\cap$ Studente[CFis] = $\varnothing$

Si lascia come esercizio la definizione delle viste per ricostruire le relazioni dello schema logico originario

Esercizio 7: confronto tra schema logico proveniente dalla traduzione diretta e schema logico ristrutturato



Esercizio 8: ristrutturare il seguente schema ...

Volo(Codice, Comp, Durata)
  foreign key: Volo[Comp] $\subseteq$ Compagnia[Nome]
  foreign key: Volo[Codice,Comp] $\subseteq$ ArrPart[Codice,Comp]

ArrPart(Codice, Comp, Arrivo, Partenza)
  foreign key: ArrPart[Arrivo] $\subseteq$ Aeroporto[Codice]
  foreign key: ArrPart[Partenza] $\subseteq$ Aeroporto[Codice]
  foreign key: ArrPart[Codice,Comp] $\subseteq$ Volo[Codice,Comp]
  chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo)
  foreign key: VoloCharter[Codice,Comp] $\subseteq$ Volo[Codice,Comp]

Aeroporto(Codice, Nome)
  foreign key: Aeroporto[Codice] $\subseteq$ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
  foreign key: LuogoAeroporto[Aeroporto] $\subseteq$ Aeroporto[Codice]
  foreign key: LuogoAeroporto[NomeCittà,NazCittà] $\subseteq$
    Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond)
  foreign key: Compagnia[Nome] $\subseteq$ Sede[Comp]

Esercizio 8: ristrutturare il seguente schema...(2)

Sede(Comp)
- foreign key: Sede[Comp] $\subseteq$ Compagnia[Nome]
- foreign key: Sede[Comp] $\subseteq$ CittàSede[Comp]
- inclusione: Sede[Comp] $\subseteq$ TelefonoComp[Comp]

CittàSede(Comp, NomeCittà, NazCittà)
- foreign key: CittàSede[Comp] $\subseteq$ Sede[Comp]
- foreign key: CittàSede[NomeCittà,NazCittà] $\subseteq$ Città[Nome,Nazione]

TelefonoComp(Numero, Comp)
- foreign key: Telefono[Comp] $\subseteq$ Sede[Comp]
- foreign key: TelefonoComp[Numero] $\subseteq$ Telefono[Numero]

Telefono(Numero)
- foreign key: Telefono[Numero] $\subseteq$ TelefonoComp[Numero]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
- foreign key: Tappa[CodVoloCharter,Comp] $\subseteq$ VoloCharter[Codice,Comp]
- foreign key: Tappa[Aeroporto] $\subseteq$ Aeroporto[Codice]

Vincolo esterno: vincolo su Ordine in Tappa

... tenendo conto delle seguenti specifiche

- Si accede spesso all'insieme dei voli charter separatamente dagli altri voli, sia per quanto riguarda la durata, sia per quanto riguarda gli aeroporti di arrivo e partenza
- Quando si accede ad un volo (charter o no) si vuole spesso conoscere tutte le proprietà di tale volo (durata, tipo aereo se volo charter, aeroporto di arrivo e aeroporto di partenza)
- Quando si accede ad una compagnia aerea si accede anche ai dati relativi alla sua sede

Esercizio 8: soluzione – ristrutturazioni

- Si accede spesso all'insieme dei voli charter separatamente dagli altri voli, sia per quanto riguarda la durata, sia per quanto riguarda gli aeroporti di arrivo e partenza:
  - **decomposizione orizzontale** di Volo in
    - VoloNonCharter
    - DatiVoloCharter
  - **decomposizione orizzontale** di ArrPart in
    - ArrPartVoloNonCharter
    - ArrPartVoloCharter
- Quando si accede ad un volo (charter o no) si vuole spesso conoscere tutte le proprietà di tale volo (durata, tipo aereo se volo charter, aeroporto di arrivo e aeroporto di partenza):
  - **accorpamento** di DatiVoloCharter e VoloCharter e, successivamente, della relazione risultante con ArrPartVoloCharter
  - **accorpamento** di VoloNonCharter e ArrPartVoloNonCharter
- Quando si accede alla compagnia si accede anche ai dati relativi alla sua sede:
  - **accorpamento** di Compagnia e CittàSede
- Si eliminano le relazioni inutili Sede e Telefono mediante **accorpamento**

Esercizio 8: soluzione – schema ristrutturato

VoloNonCharter(Codice, Comp, Durata, Arrivo, Partenza)
- foreign key: VoloNonCharter[Comp] $\subseteq$ Compagnia[Nome]
- foreign key: VoloNonCharter[Arrivo] $\subseteq$ Aeroporto[Codice]
- foreign key: VoloNonCharter[Partenza] $\subseteq$ Aeroporto[Codice]
- chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo, Durata, Arrivo, Partenza)
- foreign key: VoloCharter[Comp] $\subseteq$ Compagnia[Nome]
- foreign key: VoloCharter[Arrivo] $\subseteq$ Aeroporto[Codice]
- foreign key: VoloCharter[Partenza] $\subseteq$ Aeroporto[Codice]
- chiave: Comp, Arrivo, Partenza

Aeroporto(Codice, Nome)
- foreign key: Aeroporto[Codice] $\subseteq$ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
- foreign key: LuogoAeroporto[Aeroporto] $\subseteq$ Aeroporto[Codice]
- foreign key: LuogoAeroporto[NomeCittà, NazCittà] $\subseteq$ Città[Nome, Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond, NomeCittà, NazCittà)
- foreign key: Compagnia[NomeCittà, NazCittà] $\subseteq$ Città[Nome, Nazione]
- inclusione: Compagnia[Nome] $\subseteq$ TelefonoComp[Comp]

TelefonoCom(Numero, Comp)
- foreign key: TelefonoComp[Comp] $\subseteq$ Compagnia[Nome]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
- foreign key: Tappa[CodVoloCharter, Comp] $\subseteq$ VoloCharter[Codice, Comp]
- foreign key: Tappa[Aeroporto] $\subseteq$ Aeroporto[Codice]

Esercizio 8: soluzione – vincoli e viste

Vincoli:

- VoloNonCharter e VoloCharter sono disgiunti:

  VoloNonCharter[Codice, Comp] ∩ VoloCharter[Codice, Comp] = ∅

  VoloNonCharter[Comp, Arrivo, Partenza] ∩ VoloCharter[Comp, Arrivo, Partenza] = ∅

- vincolo su Ordine in Tappa

Viste per ricostruire le relazioni dello schema originario:

view Volo = $PROJ_{Codice,Comp,Durata}(VoloNonCharter) \cup PROJ_{Codice,Comp,Durata}(VoloCharter)$

view ArrPart = $PROJ_{Codice,Comp,Arrivo,Partenza}(VoloNonCharter) \cup PROJ_{Codice,Comp,Arrivo,Partenza}(VoloCharter)$

view Sede = $PROJ_{Nome}(Compagnia)$

view CompagniaOriginaria = $PROJ_{Nome,\ AnnoFond}(Compagnia)$

view CittàSede = $PROJ_{Nome,NomeCittà,NazCittà}(Compagnia)$

view Telefono = $PROJ_{Numero}(TelefonoComp)$

Esercizio 8: confronto tra schema logico proveniente dalla traduzione diretta e schema logico ristrutturato

Schema logico da traduzione diretta

Volo(Codice, Comp, Durata)
- foreign key: Volo[Comp] ⊆ Compagnia[Nome]
- foreign key: Volo[Codice,Comp] ⊆ ArrPart[Codice,Comp]

ArrPart(Codice, Comp, Arrivo, Partenza)
- foreign key: ArrPart[Arrivo] ⊆ Aeroporto[Codice]
- foreign key: ArrPart[Partenza] ⊆ Aeroporto[Codice]
- foreign key: ArrPart[Codice,Comp] ⊆ Volo[Codice,Comp]
- chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo)
- foreign key: VoloCharter[Codice,Comp] ⊆ Volo[Codice,Comp]

Aeroporto(Codice, Nome)
- foreign key: Aeroporto[Codice] ⊆ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
- foreign key: LuogoAeroporto[Aeroporto] ⊆ Aeroporto[Codice]
- foreign key: LuogoAeroporto[NomeCittà,NazCittà] ⊆ Città[Nome,Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond)
- foreign key: Compagnia[Nome] ⊆ Sede[Comp]

Sede(Comp)
- foreign key: Sede[Comp] ⊆ Compagnia[Nome]
- foreign key: Sede[Comp] ⊆ CittàSede[Comp]
- inclusione: Sede[Comp] ⊆ TelefonoComp[Comp]

CittàSede(Comp, NomeCittà, NazCittà)
- foreign key: CittàSede[Comp] ⊆ Sede[Comp]
- foreign key: CittàSede[NomeCittà,NazCittà] ⊆ Città[Nome,Nazione]

TelefonoComp(Numero, Comp)
- foreign key: Telefono[Comp] ⊆ Sede[Comp]
- foreign key: TelefonoComp[Numero] ⊆ Telefono[Numero]

Telefono(Numero)
- foreign key: Telefono[Numero] ⊆ TelefonoComp[Numero]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
- foreign key: Tappa[CodVoloCharter,Comp] ⊆ VoloCharter[Codice,Comp]
- foreign key: Tappa[Aeroporto] ⊆ Aeroporto[Codice]

Schema logico ristrutturato

VoloNonCharter(Codice, Comp, Durata, Arrivo, Partenza)
- foreign key: VoloNonCharter[Comp] ⊆ Compagnia[Nome]
- foreign key: VoloNonCharter[Arrivo] ⊆ Aeroporto[Codice]
- foreign key: VoloNonCharter[Partenza] ⊆ Aeroporto[Codice]
- chiave: Comp, Arrivo, Partenza

VoloCharter(Codice, Comp, TipoAereo, Durata, Arrivo, Partenza)
- foreign key: VoloCharter[Comp] ⊆ Compagnia[Nome]
- foreign key: VoloCharter[Arrivo] ⊆ Aeroporto[Codice]
- foreign key: VoloCharter[Partenza] ⊆ Aeroporto[Codice]
- chiave: Comp, Arrivo, Partenza

Aeroporto(Codice, Nome)
- foreign key: Aeroporto[Codice] ⊆ LuogoAeroporto[Aeroporto]

LuogoAeroporto(Aeroporto, NomeCittà, NazCittà)
- foreign key: LuogoAeroporto[Aeroporto] ⊆ Aeroporto[Codice]
- foreign key: LuogoAeroporto[NomeCittà, NazCittà] ⊆ Città[Nome, Nazione]

Città(Nome, Nazione, NumAbitanti)

Compagnia(Nome, AnnoFond, NomeCittà, NazCittà)
- foreign key: Compagnia[NomeCittà, NazCittà] ⊆ Città[Nome, Nazione]
- inclusione: Compagnia[Nome] ⊆ TelefonoComp[Comp]

TelefonoCom(Numero, Comp)
- foreign key: TelefonoComp[Comp] ⊆ Compagnia[Nome]

Tappa(CodVoloCharter, Comp, Aeroporto, Ordine)
- foreign key: Tappa[CodVoloCharter, Comp] ⊆ VoloCharter[Codice, Comp]
- foreign key: Tappa[Aeroporto] ⊆ Aeroporto[Codice]

Esercizio 9: ristrutturare il seguente schema ...

Officina(Nome, NumDip, Indirizzo)
- foreign key: Officina[Nome] $\subseteq$ Dirige[Officina]
- inclusione: Officina[Nome] $\subseteq$ Lavora[Officina]

Persona(CodFis, Indirizzo)

Direttore(CodFis, Età, AnniAnz)
- foreign key: Direttore[CodFis] $\subseteq$ Persona[CodFis]
- foreign key: Direttore[CodFis] $\subseteq$ Dirige[Direttore]

Dipendente(CodFis, AnniAnz)
- foreign key: Dipendente[CodFis] $\subseteq$ Persona[CodFis]
- inclusione: Dipendente[CodFis] $\subseteq$ Lavora[Dipendente]

Dirige(Officina, Direttore)
- foreign key: Dirige[Officina] $\subseteq$ Officina[Nome]
- foreign key: Dirige[Direttore] $\subseteq$ Direttore[CodFis]
- chiave: Direttore

Lavora(Officina, Dipendente, AnniServizio)
- foreign key: Lavora[Officina] $\subseteq$ Officina[Nome]
- foreign key: Lavora[Dipendente] $\subseteq$ Dipendente[CodFis]

TelPer(CodFis, Telefono)
- foreign key: TelPer[CodFis] $\subseteq$ Persona[CodFis]
- foreign key: TelPer[Telefono] $\subseteq$ Telefono[Numero]

Esercizio 9: ristrutturare il seguente schema...(parte 2)

Telefono(Numero)
inclusione: Telefono[Numero] $\subseteq$ TelPer[Telefono]

Veicolo(Targa, Modello, Tipo, AnnoImm)
foreign key : Veicolo[Targa] $\subseteq$ Possiede[Veicolo]

Possiede(Veicolo, Proprietario)
foreign key: Possiede[Veicolo] $\subseteq$ Veicolo[Targa]
foreign key: Possiede[Proprietario] $\subseteq$ Persona[CodFis]

Riparazione(Codice, Officina, OraAcc, DataAcc)
inclusione: Riparazione[Officina] $\subseteq$ Officina[Nome]
foreign key: Riparazione[Codice,Officina] $\subseteq$ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
foreign key: Relativa[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]
foreign key: Relativa[Veicolo] $\subseteq$ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
foreign key: Terminata[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]

Vincoli esterni:

- riconsegna dopo accettazione
- vincolo che lega Officina[NumDip] alle istanze in Lavora
- vincolo su AnniAnz di Direttore e Dipendente derivante dall'eliminazione ISA

... tenendo conto delle seguenti specifiche

- Quando si accede ai direttori, interessano anche tutti i dati relativi all'officina che dirigono e viceversa, quando si accede alle officine, interessano anche i dati relativi all'età e alla anzianità del loro direttore
- Alle persone dipendenti si accede spesso separatamente rispetto alle persone direttori
- Quando si accede ai dipendenti interessano anche i dati relativi all'indirizzo

Esercizio 9: soluzione – ristrutturazioni

- Quando si accede ai direttori, interessano anche tutti i dati relativi all'officina che dirigono e viceversa quando si accede alle officine, interessano anche tutti i dati relativi al loro direttore:
  - **accorpamento** di Direttore e Dirige e, successivamente, della relazione risultante con Officina
- Ai dipendenti si accede spesso separatamente rispetto ai direttori:
  - **decomposizione orizzontale** di Persona in dipendenti e non
- Quando si accede ai dipendenti interessano anche i loro dati anagrafici:
  - **accorpamento** delle relazioni Dipendente e PersonaDipendente
- Si elimina la relazione inutile Telefono mediante **accorpamento**

Esercizio 9: soluzione – schema ristrutturato

OfficinaDirettore(Nome, NumDip, Indirizzo, Direttore, EtaDir, AnniAnzDir)
chiave: Direttore
inclusione: Officina[Nome] $\subseteq$ Lavora[Officina]

PersonaNonDip(CodFis, Indirizzo)

Dipendente(CodFis, AnniAnz, Indirizzo)
inclusione: Dipendente[CodFis] $\subseteq$ Lavora[Dipendente]

Lavora(Officina, Dipendente, AnniServizio)
foreign key: Lavora[Officina] $\subseteq$ OfficinaDirettore[Nome]
foreign key: Lavora[Dipendente] $\subseteq$ Dipendente[CodFis]

TelPer(CodFis, Telefono)

Veicolo(Targa, Modello, Tipo, AnnoImm)

Possiede(Veicolo, Proprietario)
foreign key: Possiede[Veicolo] $\subseteq$ Veicolo[Targa]

Riparazione(Codice, Officina, OraAcc, DataAcc,)
inclusione: Riparazione[Officina] $\subseteq$ OfficinaDirettore[Nome]
foreign key: Riparazione[Codice,Officina] $\subseteq$ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
foreign key: Relativa[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]
foreign key: Relativa[Veicolo] $\subseteq$ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
foreign key: Terminata[Codice,Officina] $\subseteq$ Riparazione[Codice,Officina]

Esercizio 9: soluzione – vincoli e viste

Vincoli:

- PersonaNonDip e Dipendente sono disgiunti:
  - $\text{PersonaNonDip}[\text{CodFis}] \cap \text{Dipendente}[\text{CodFis}] = \varnothing$
- Vincoli risultanti dai vincolo di foreign key verso Persona
  - $\text{Officina}[\text{Direttore}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
  - $\text{Veicolo}[\text{Proprietario}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
  - $\text{TelPer}[\text{CodFis}] \subseteq \text{PersonaNonDip}[\text{CodFis}] \cup \text{Dipendente}[\text{CodFis}]$
- Vincoli esterni:
  - riconsegna dopo accettazione
  - vincolo che lega Officina[NumDip] alle istanze in Lavora
  - vincolo su Officina[AnniAnzDir] e Dipendente[AnniAnz] derivante dall'eliminazione ISA

Viste per ricostruire le relazioni dello schema originario:

view Persona = $\text{PersonaNonDip} \cup \text{PROJ}_{\text{CodFis,Indirizzo}}(\text{Dipendente})$

view OfficinaOrig = $\text{PROJ}_{\text{Nome, NumDip, Indirizzo}}(\text{Officina})$

view Direttore = $\text{PROJ}_{\text{Direttore, EtaDir, AnniAnzDir}}(\text{Officina})$

view Dirige = $\text{PROJ}_{\text{Nome, Direttore}}(\text{Officina})$

view Telefono = $\text{PROJ}_{\text{Telefono}}(\text{TelPer})$

Esercizio 9: confronto tra schema logico proveniente dalla traduzione diretta e schema logico ristrutturato

Schema logico da traduzione diretta

Officina(Nome, NumDip, Indirizzo)
- foreign key: Officina[Nome] ⊆ Dirige[Officina]
- inclusione: Officina[Nome] ⊆ Lavora[Officina]

Persona(CodFis, Indirizzo)

Direttore(CodFis, Età, AnniAnz)
- foreign key: Direttore[CodFis] ⊆ Persona[CodFis]
- foreign key: Direttore[CodFis] ⊆ Dirige[Direttore]

Dipendente(CodFis, AnniAnz)
- foreign key: Dipendente[CodFis] ⊆ Persona[CodFis]
- inclusione: Dipendente[CodFis] ⊆ Lavora[Dipendente]

Dirige(Officina, Direttore)
- foreign key: Dirige[Officina] ⊆ Officina[Nome]
- foreign key: Dirige[Direttore] ⊆ Direttore[CodFis]
- chiave: Direttore

Lavora(Officina, Dipendente, AnniServizio)
- foreign key: Lavora[Officina] ⊆ Officina[Nome]
- foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]

TelPer(CodFis, Telefono)
- foreign key: TelPer[CodFis] ⊆ Persona[CodFis]
- foreign key: TelPer[Telefono] ⊆ Telefono[Numero]

Telefono(Numero)
- inclusione: Telefono[Numero] ⊆ TelPer[Telefono]

Veicolo(Targa, Modello, Tipo, AnnoImm)
- foreign key : Veicolo[Targa] ⊆ Possiede[Veicolo]

Possiede(Veicolo, Proprietario)
- foreign key: Possiede[Veicolo] ⊆ Veicolo[Targa]
- foreign key: Possiede[Proprietario] ⊆ Persona[CodFis]

Riparazione(Codice, Officina, OraAcc, DataAcc)
- inclusione: Riparazione[Officina] ⊆ Officina[Nome]
- foreign key: Riparazione[Codice,Officina] ⊆ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
- foreign key: Relativa[Codice,Officina] ⊆ Riparazione[Codice,Officina]
- foreign key: Relativa[Veicolo] ⊆ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
- foreign key: Terminata[Codice,Officina] ⊆ Riparazione[Codice,Officina]

Schema logico ristrutturato

OfficinaDirettore(Nome,NumDip,Indirizzo,Direttore,EtaDir,AnniAnzDir)
- chiave: Direttore
- inclusione: Officina[Nome] ⊆ Lavora[Officina]

PersonaNonDip(CodFis, Indirizzo)

Dipendente(CodFis, AnniAnz, Indirizzo)
- inclusione: Dipendente[CodFis] ⊆ Lavora[Dipendente]

Lavora(Officina, Dipendente, AnniServizio)
- foreign key: Lavora[Officina] ⊆ OfficinaDirettore[Nome]
- foreign key: Lavora[Dipendente] ⊆ Dipendente[CodFis]

TelPer(CodFis, Telefono)

Veicolo(Targa, Modello, Tipo, AnnoImm)

Possiede(Veicolo, Proprietario)
- foreign key: Possiede[Veicolo] ⊆ Veicolo[Targa]

Riparazione(Codice, Officina, OraAcc, DataAcc,)
- inclusione: Riparazione[Officina] ⊆ OfficinaDirettore[Nome]
- foreign key: Riparazione[Codice,Officina] ⊆ Relativa[Codice,Officina]

Relativa(Codice, Officina, Veicolo)
- foreign key: Relativa[Codice,Officina] ⊆ Riparazione[Codice,Officina]
- foreign key: Relativa[Veicolo] ⊆ Veicolo[Targa]

Terminata(Codice, Officina, OraRic, DataRic)
- foreign key: Terminata[Codice,Officina] ⊆ Riparazione[Codice,Officina]

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è *v*. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	2000
C1	A3	1000
C2	A2	2000
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	2000
C3	A6	2500

Cosa succede se il valore dell'appartamento A2 cambia da 2000 a 1800?

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	1800
C1	A3	1000
C2	A2	1800
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	1800
C3	A6	2500

Cosa succede se il valore dell'appartamento A2 cambia da 2000 a 1800?

Occorre cambiare il valore in tutte le tuple in cui compare il valore A2 nell'attributo Appartamento.

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	2000
C1	A3	1000
C2	A2	2000
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	2000
C3	A6	2500

Cosa succede se C2 abbandona l'agenzia?

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	2000
C1	A3	1000
C2	A2	2000
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	2000
C3	A6	2500

Cosa succede se C2 abbandona l'agenzia? Vengono persi i dati relativi al valore degli appartamenti A4 e A5

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	2000
C1	A3	1000
C2	A2	2000
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	2000
C3	A6	2500

Si può inserire il valore di un appartamento se quest'ultimo non interessa almeno un cliente?

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v. Consideriamo ad esempio questa istanza della relazione:

Cliente	Appartamento	Valore
C1	A1	1000
C1	A2	2000
C1	A3	1000
C2	A2	2000
C2	A4	4000
C2	A5	2000
C2	A6	2500
C3	A2	2000
C3	A6	2500

Si può inserire il valore di un appartamento se quest'ultimo non interessa almeno un cliente?

No, non possiamo inserire NULL nell'attributo cliente.

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v.

La tabella soffre quindi di alcune "anomalie":

1. c'è ridondanza, perché il valore di un appartamento è ripetuto in diverse tuple della tabella
2. proprio a causa della ridondanza, l'aggiornamento rischia di essere molto costoso computazionalmente
3. il fatto che il valore di un appartamento sia associato ai clienti che sono interessati all'appartamento rischia di far perdere l'informazione sul valore, nel caso in cui i clienti interessati a quell'appartamento sono eliminati dalla base di dati.

Normalizzazione e denormalizzazione

Intuitivamente, diciamo che una tabella R è normalizzata se ogni suo attributo dipende funzionalmente dalla chiave della tabella. In altre parole, se non ci sono "legami nascosti" nella tabella, ovvero legami che non ricadano nel rapporto tra chiave e attributi non chiave.

Nel caso della tabella Interesse, la chiave è chiaramente <cliente,appartamento>, ma l'attributo "valore" dipende funzionalmente dal solo attributo appartamento, che non è chiave (è una parte della chiave). Ne segue che l'attributo valore non dipende funzionalmente dalla chiave e quindi nella tabella esiste un legame tra appartamento e valore che è "nascosto", nel senso che non è testimoniato dalla chiave. Quindi la tabella Interesse non è normalizzata (si dice che è denormalizzata) e da questo derivano le anomalie discusse in precedenza.

Normalizzazione e denormalizzazione

Consideriamo la seguente tabella, appartenente ad una base di dati gestita da un'agenzia di immobili:

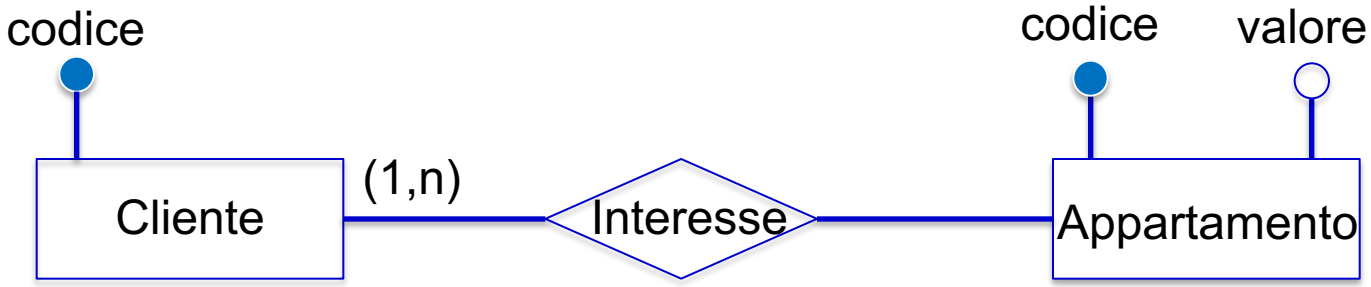
Interesse(cliente, appartamento, valore)

in cui ogni tupla <c,a,v> rappresenta il fatto che il cliente c ha espresso interesse verso l'appartamento a il cui valore per metro quadrato è v.

Fermo restando le anomalie discusse nella slide precedente, osserviamo che se l'accesso ad un cliente richiede sempre di conoscere il valore degli appartamenti ai quali il cliente è interessato, la denormalizzazione può diventare vantaggiosa per l'esecuzione della operazione di accesso ai clienti, in particolare se il valore degli appartamenti cambia raramente.

Normalizzazione e denormalizzazione

Riferendoci all'esempio precedente, la progettazione concettuale avrebbe prodotto lo schema:



tradotto poi in relazionale in queste tre tabelle normalizzate:

Cliente(codice)

inclusione Cliente[codice] $\subseteq$ Interesse[cliente]

Interesse(<u>cliente</u>, <u>appartamento</u>)

foreign key Interesse[cliente] $\subseteq$ Cliente[codice]

foreign key Interesse[appartamento] $\subseteq$ Appartamento[valore]

Appartamento(<u>codice</u>,valore)

Osservazione: le tabelle che provengono dalla traduzione diretta di uno schema concettuale di qualità sono sempre normalizzate!

Normalizzazione e denormalizzazione

Se però dalle indicazioni di progetto e dalla conoscenza delle operazioni giungiamo alla conclusione che ogni volta che si accede a Interesse si vuole conoscere il valore degli appartamenti acceduti, e che il valore degli appartamenti cambia raramente, allora nella fase di ristrutturazione dello schema relazionale si possono accorpare le tabelle (accoppiate lascamente) Interesse e Appartamento, evitando così join costosi. L'accorpamento in questione è un esempio di accorpamento per denormalizzazione visto prima. Lo schema risultante sarebbe:

Cliente(codice)

inclusione Cliente[codice] $\subseteq$ Interesse[cliente]

Interesse(cliente, appartamento,valore)

foreign key Interesse[cliente] $\subseteq$ Cliente[codice]

foreign key Interesse[appartamento,valore] $\subseteq$ Appartamento[codice,valore]

Appartamento(codice,valore)

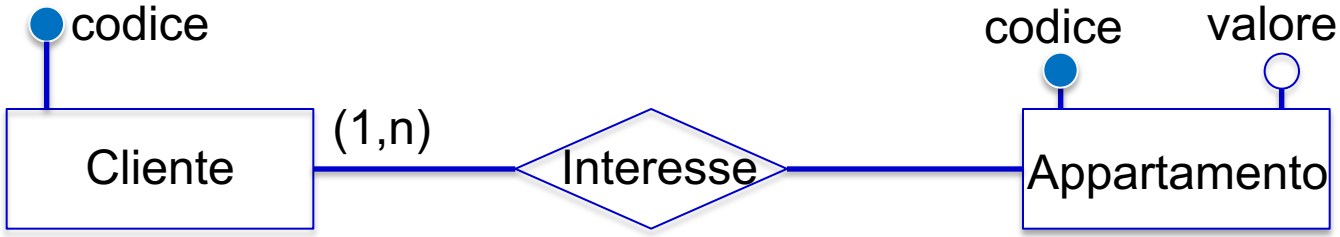
A questo punto possiamo anche applicare l'accorpamento per eliminare la relazione Cliente, visto che ci sono le condizioni di accoppiamento tra Cliente ed Interesse. Se lo facciamo otteniamo lo schema finale mostrato nella prossima slide.

Normalizzazione e denormalizzazione

Interesse(cliente, appartamento,valore)

foreign key Interesse[appartamento,valore] $\subseteq$ Appartamento[codice,valore]

Appartamento(codice,valore)



Partendo quindi dallo schema concettuale qui sopra e dalle tre tabelle normalizzate ottenute con la traduzione diretta, abbiamo considerato il carico applicativo, quindi il modo in cui l'applicazione usa le tabelle, ed abbiamo ottenuto lo schema relazionale finale non normalizzato.

Questo esempio ribadisce ciò che abbiamo già visto prima: in fase di ristrutturazione dello schema logico, qualora sia vantaggioso farlo, si possono denormalizzare le tabelle (mediante accorpamenti per denormalizzazione), in particolare al fine di evitare join in query frequenti e importanti.